FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

A secure global time base for time triggered systems

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Teschnische Informatik

eingereicht von

Haris Isakovic
Matrikelnummer 0325697

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter PUSCHNER
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Christian El-Salloum

Wien, 26.08.2011

(Unterschrift Verfasserin) (Unterschrift Betreuung)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A secure global time base for time triggered systems

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Computer Science

by

Haris Isakovic
Registration Number 0325697

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter PUSCHNER
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Christian El-Salloum

Vienna, 26.08.2011

(Signature of Author) (Signature of Advisor)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Haris Isakovic
Hagenmüllergasse 27-33/105, 1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Acknowledgements

Without the people around me the completion of this thesis would not have been possible.

I owe my deepest gratitude to Univ.Ass. Dipl.-Ing. Dr.techn. Christian El-Salloum, who made the work on this thesis possible. He supported me as a mentor and a colleague throughout this endeavour and showed an excellent competence by doing so.

I would like to thank Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Puschner for evaluating my work and providing me with the most helpful insights in the process.

A special thanks to my dear friend and colleague Emir Causevic. We shared this experience from the beginning and he provided me with the invaluable help, support, and most of all an unforgeable time.

To all the collogues at the Real-Time Systems Group of the University of Vienna I would like to express my sincere gratitude for their assistance and friendship.

I am indebted to my beloved girlfriend Nerma as she encouraged and motivated me during all this time.

Most of all I would like to thank my family: my mother Enisa, my father Sefik nad my sister Nerma. Their suport has been essential all these years and it is an honour to be able to thank them like this.

Abstract

Time-Triggered protocols provide high dependability and guaranteed timeliness and are present in many distributed real-time applications today. They provide various services such as clock synchronisation, membership, redundancy management etc. The ever growing demand for dependable real-time systems imposes new requirements on these communication protocols. One of the open challenges is security.Current implementations of time-triggered communication protocols are not focused on security and their protection against malicious attacks is weak or non-existing.

Time-triggered systems are based on a consistent notion of time. The functionality of all essential services in these systems depends on this time, therefore it is of vital importance that the global time is secured against malicious attacks (i.e. unauthorized modification).

The main task of this thesis is to design a security layer for the Time-Triggered Architecture (TTA) with a focus on implementing a secure and fault tolerant clock synchronisation algorithm. The approach consists of a platform-independent security layer realized on top of the existing clock synchronisation algorithm provided by the underlying time-triggered communication protocol. In this thesis we use Time-Triggered Ethernet as an implementation platform.

Our security layer protects the global time from many different kinds of malicious attacks like the fabrication, modification, replay, delay or speed up of clock synchronization messages. Our approach is based on an interplay of asymmetric and symmetric ciphers, and provides a high level of security while keeping the resource overhead low.

The feasibility of our approach is demonstrated by carefully selected experiments, that show how the time base of unprotected standard time-triggered protocols can be attacked, and how our security layer reliably detects such attacks. Furthermore, various tests have been conducted in an experimental setup in order to measure the computational overhead and the general usage of system resources.

Kurzfassung

Zeitgesteuerte Protokolle bieten hohe Zuverlässigkeit und garantiertes zeitliches Verhalten und werden heutzutage in vielen verteilten Echtzeitanwendungen eingesetzt. Sie bieten verschiedene Dienste wie z.B. Uhrensynchronisation, Membership, Redundanz-Management etc. Der ständig steigende Bedarf nach zuverlässigen Echtzeitsystemen stellt neue Anforderungen an die Kommunikationsprotokolle. Eine der noch offenen Herausforderungen ist die Informationssicherheit (auf English *Security*). Die aktuellen Implementierungen von zeitgesteuerten Kommunikationsprotokollen sind nicht auf Informationssicherheit ausgerichtet und ihr Schutz gegen bösartige Attacken ist nicht ausreichend.

Zeitgesteuerte Systeme benötigen eine konsistente Sicht der Zeit im gesamten System. Die Funktionalität aller essentiellen Dienste in diesen Systemen hängt von der Zeit ab. Deshalb ist es von entscheidender Bedeutung, dass die globale Zeit gegen bösartige Attacken gesichert ist.

Die Hauptaufgabe dieser Diplomarbeit ist die Entwicklung einer Sicherheitsschicht für die Time-Triggered Architecture (TTA) mit dem Schwerpunkt auf einer sicheren und fehlertoleranten Uhrensynchronisation. Der Ansatz besteht aus einer plattformunabhängigen Sicherheitsschicht oberhalb des bestehenden Uhrensynchronisationsalgorithmus. In dieser Arbeit verwenden wir Time-Triggered Ethernet (TTE) als eine Implementierungsplattform.

Unsere Sicherheitsschicht schützt die globale Zeitbasis vor verschiedenen bösartigen Angriffsarten, z.B. der Fälschung, der Modifikation, der Wiederholung, oder der Verzögerung oder der Beschleunigung der Uhrensynchronisationsnachrichten. Unser Ansatz basiert auf einem Wechselspiel von asymmetrischen und symmetrischen Verschlüsselungsverfahren, und bietet ein hohes Sicherheitsniveau, mit geringem Ressourcenverbrauch.

Die Realisierbarkeit des Ansatzes wird durch sorgfältig ausgewählte Experimente demonstriert. Die Experimente zeigen, wie die Zeitbasis des ungeschützten normalen zeitgesteuerte Protokolls angegriffen werden kann, und wie unsere Sicherheitsschicht solche Angriffe zuverlässig erkennt. Darüber hinaus werden verschiedene Tests in einer Versuchsanordnung durchgeführt, um den Rechenaufwand und den allgemeinen Ressourcenverbrauch zu messen.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

This thesis explores a special kind of computer systems called embedded systems [1]. This special class of computers systems are a computer system integrated in a larger system with strictly defined tasks. In most cases they are connected with a mechanical subsystem, user interface and communication system. The use of embedded systems is extensive, they can be found in every modern kitchen appliance, almost every car has one, and they control the flight control systems in an airplane. Embedded systems are used in many applications and every day more and more people depend their correct service. They are expected to control both non critical systems and safety critical systems with great reliability. Therefore they have to be protected against both, accidental and malicious faults. Each fault in a safety critical system can cause a large safety risk and endanger human lives.

Although they operate as individual units, providing in most cases only few, their power is the connectivity in a larger distributed system. The communication system used for the coordination of the single embedded computers in the network must be reliable and secure. A failure in the communication infrastructure may lead to the failure of the entire system.

One of the communication methods used in large number of embedded applications is time-triggered communication [2]. The strength of the time triggered communication is its determinism. To establish determinism, the communication protocol implements a clock synchronization algorithm to establish a global notion of time for all members in the network. Only then the communication and the execution of tasks can be considered to be deterministic. This is the most important requirement on every time triggered system. If the clock synchronization algorithm is in any way dysfunctional the determinism is lost, and the whole system is compromised. The clock synchronization can be disabled with a malicious attack and doing so, endangers the entire system. If the system is safety critical, the extent of the damage could be catastrophic with even human casualties (i.e. car failure while driving).

The goal of this thesis is to isolate potential weak spots of the global time base of time-triggered systems and to provide a reliable security policy which will be able to protect against various any attacks against the global time base. This security policy integrated with other security methods will constitute a security platform for embedded time-triggered systems.

This security policy is implemented as a security overlay for current clock synchronization algorithms. It implements all security techniques and tools needed for the secure clock synchronization without changing the actual algorithm clock synchronization.

1.1 Contribution

In the scope of this thesis a security layer is developed which provides a secure global time base for time triggered systems. The secure layer is implemented on top the Time Triggered Ethernet [3] protocol.

First, the potential security threats are identified. The security properties violated by these threats are identified requirements for a security model. Based on this model an algorithm is developed for the protection of the global time base. The design also considers the cryptographic methods required to achieve the security properties.

For the implementation of the design a development environment is built. It provides all necessary hardware and software for the Time Triggered Ethernet system. The development environment is designed as a generic platform and can be used for other research projects, in the area of the computer communication.

Finally several experiments are performed to test the implemented security layer on Time Triggered Ethernet. The achieved results show significant improvement of the security of the global time base in Time Triggered Ethernet.

1.2 Outline

The thesis is structured as follows:

- Chapter 3 describes basic terms and concepts in the field of real-time systems, distributed systems, Time Triggered Ethernet Protocol and security of computer systems. The chapter gives an introduction on time-triggered communication and cryptography.

- Chapter 2 gives a short overview of two current implementations of secure clock synchronization and the description of various research projects which can be integrated with the main concepts and the results of this thesis to create a more complete security platform for the Time-Triggered Architecture.

- Chapter 4 describes the system model for the secure clock synchronization protocol. In this chapter malicious threats on the global time base of time-triggered systems are identified and measures to counteract potential attacks are proposed. The secure clock synchronization protocol is designed as security layer on top of the existing time triggered Ethernet protocol. It uses cryptographic tools to achieve authentication and other security properties.

- Chapter 5 provides an overview of the development environment and all its components. The hardware, software and the tools for development, debugging and evaluation are described. It also gives a detailed description of the implementation of the model for Time Triggered Ethernet. This includes the description of the configuration data and the algorithms used to implement the secure clock synchronization algorithm.

- Chapter 6 describes the experiments performed for the evaluation of the secure clock synchronization protocol implemented on Time Triggered Ethernet. The experiment shows that the global time is indeed vulnerable with respect to malicious attacks and that these attacks can be prevented, with the security model described in the Chapter 4.
- Chapter 7 contains the final words on the thesis and a discussion of the results acquired from the experiments.

CHAPTER 2

Related Work

The protection of clock synchronization protocols against malicious attacks is the concern and the research field of many scientists over the last few years. There are several clock synchronization protocols which implement security mechanisms for the protection against malicious attacks. Although many of these protocols are designed for embedded systems they provide no efficient solution for securing the clock synchronization in a time triggered architecture [2]. In this chapter two secure clock synchronization protocols are described: a clock synchronization of the IEEE 1588 standard [4], and a secure clock synchronization protocol for wireless networks [5]. Also a security platform designed for time-triggered systems is described and the integration of this thesis with other research projects to create this kind of a security platform for the time-triggered systems.

2.1 IEEE 1588

The IEEE 1588 is a standard for network clock synchronization also called Precision Time Protocol (PTP). It is created for networked measurements and control systems, telecom applications and industrial applications. The main task of the protocol is to provide clock synchronization in distributed systems. The protocol operates on the master/slave [1] principle. For establishment of the necessary relationships it uses the Best Master Clock (BMC) algorithm.This algorithm provides division of masters throughout the network so that each part of the network that has no PTP connection with the rest of the network has one clock master. The main time source for the entire network is named a grandmaster clock. All clocks in the network first exchange information about their stability and accuracy and then the master clock is chosen. After the master is chosen, the rest of nodes have the role of a slave nodes. The precision of the clock synchronization of the PTP can be achieved in nanosecond ranges on packet oriented networks.Beside the master and slave clocks the PTP has also transparent clocks. The task of these clocks is to serve as switches and to maintain the precision of the clock synchronization.

The most of the applications using PTP have low security requirements, but still there are also applications which require high security requirements on clock synchronization. These

[1]Communication protocol where single device called master controls other devices called slaves.

applications come mostly from the field of the industrial applications. The highest threats on PTP systems represent so called man-in-the-middle replay attacks. These attacks use weakness of the systems in open networks. The attacker is able to access the lines of the network to connect a computer, which is able to cause faults in the clock synchronization systems. The security protocol for PTP is based on following security mechanisms [4]:

- A message authentication code to verify that the message is not modified in any way.
- Message counters are implemented to prevent replay of the messages.This is resending of messages and thus inserting the false data in for the certain time instant.
- For the acceptance of new members in the network and establishing their authenticity, a challenge-response mechanism is implemented. This mechanism is also used to refresh existing trust relations.

The communication of the security protocol is implemented with security associations (SAs). The SA provides a necessary infrastructure for the security protocol. It contains a source and a destination information (e.g. port, address), a key, a lifetime identification number, and a replay counter. The SA is valid in only one direction. Each SA has only one source, but it can have multiple destinations. The sender node is responsible for creating a SA. After the SA is created, it is transmitted to the receiver. The sender decides whether it will create one SA for multiple receivers or one SA for each receiver. The key of the SA is shared among all members which use that SA and it serves as a secret key for message authentication algorithm. The lifetime identification number, together with the replay counter implement a protection against replay attacks.

The protocol requires an additional system for the distribution of the secret. It provides protection against message modification and message replay, but it fails to provide accountability and traceability of the actions in the network. It also provides no protection against gradual delay or speed up, of the synchronization messages.

2.2 Secure Time Synchronization Protocols for Sensor Networks

A wireless sensor network is a network of embedded systems serving as sensors for observing real-world phenomena. That small computational units are highly mobile devices with large diversity of use, due to their flexibility and robustness. Again, their lack of computational power makes them unsuitable for the implementation of usual clock synchronization protocols.In the following text one of the approaches for the secure time synchronization in wireless sensor networks is described [5].The protocol is a variation of a clock synchronization protocol, specially designed for sensor networks, with the additional security layer. The security layer provides protection of the sensor network clock synchronization protocol against external and internal attackers. The external attacker is a subject which intercepts a communication among nodes and executes malicious attack to deny correct functionality. The internal attacker is a subject, which controls one or several nodes in the network, and trough that malicious nodes create faulty conditions in the system.The protection against external attacks can be achieved with the use of message authentication. The internal attacks are more difficult. Here, the attacker can use compromised nodes to attack clock synchronization. If the attacker gets an access to the sensor node, it also gets the access to its secret keys. This is why these networks require a security scheme which can timely detect an attack.

The clock synchronization protocols for sensor networks are based on a so called single-hop pairwise synchronization.The single-hop pairwise synchronization is a technique to synchronize two neighbour nodes. The sensor networks are normally very dense and each node has several neighbours. These properties are used to implement the security scheme, which uses neighbour nodes to verify other node in its reach and detect potential attacks. The idea hides behind a fact that wireless networks use broadcast type of communication. Every node can monitor the traffic of its neighbours, including synchronization messages between synchronization pairs. Each sensor node can easily check if the synchronization of the neighbour node is correct. The implementation of the security protocol described in the research article from Li, Yanfei, Wen and Chen [5] the secure clock is achieved by combining a reference broadcast synchronization (RBS) with the sender-receiver model for single-hop pairwise clock synchronization. The execution of the protocol is implemented in three phases:

- a level discovery phase,
- a synchronization phase, and
- a verification phase.

In the level discovery phase the sensor nodes are organized in a hierarchy with a root at a base station. The base station is a computer system which provides a synchronization reference and serves as a collector of a sensor data. In the hierarchy two connected nodes are called neighbours. The nodes at level 1 are synchronized from the base station and all other nodes have verifier nodes, which control the synchronization. The synchronization phase comprises a synchronization process between each node and its parent. The two nodes synchronized from the same parent node are called siblings. They can serve to each other as verifiers and this process is called verification phase.

The method presented above is developed for wireless networks with single-hop synchronization where the principle of operation is completely different from the time-triggered networks where the medium of communication is wire and the central master synchronization algorithm is used.

2.3 Security Platforms for Time Triggered Systems

A security platform for time-triggered systems as described in the work of Wasicek [6] merges several techniques to ensure the protection against malicious threats.The platform contains a secure global time base,a message authentication for the application-level communication, and secure start-up service for a system. The techniques used to implement this platform use features specific for time-triggered systems, with cryptographic algorithms to provide needed security properties.

One of the security techniques, which considerably benefits from the properties of the time-triggered environment, is the TESLA (Timed Efficient Stream Loss-tolerant Authentication) protocol [7]. TESLA represents the message authentication technique for broadcast communication. In the work of Causevic [8] the TESLA is used to achieve a secure group communication among the members of a time-triggered network. The method is very convenient for the secure application-level communication.

For a correct execution a TESLA based authentication algorithm requires a reliable and secure clock synchronization algorithm like the one proposed in this thesis.

CHAPTER **3**

Basic Concepts

This chapter describes basic definitions and concepts on distributed real time systems and real-time communication introduced mainly by Kopetz in [1]. It also describes security concepts and mechanisms used for the implementation. The description of the basic concepts has the goal of familiarizing the reader with the topics used in the latter chapters of this thesis.

3.1 Distributed Real-Time Systems

If the functionality of a single application depends on the behaviour of several systems in a network and the communication between these systems, then these systems form a distributed system. If the functionality of such system depends not only on the correct behaviour in the value domain of all subsystems but also on the time accurate execution of tasks and a time accurate transmission of messages we say that the system operates under constraint of real-time and as such is called a Distributed Real-Time System.These systems operate under a strict schedule and even the slightest timing delay could cause faults or failures. It is therefore of essential importance that the timeliness guarantees stay valid.

3.1.1 Characteristics of a real-time system

A Distributed real-time system is set of computer systems where the fundamental unit is a real-time computer system. [1]

> *A real-time computer system is a computer system in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instant at which these results are produced.*

The real-time computer system interacts with other components through series of interfaces forming a larger system called a real-time system. Together with the components for communication with the user and components for the control of the environment, the real-time computer system forms a larger system, a real-time system.A **man-machine interface** provides connection between real-time computer and a user interface. The user interface has two basic types the

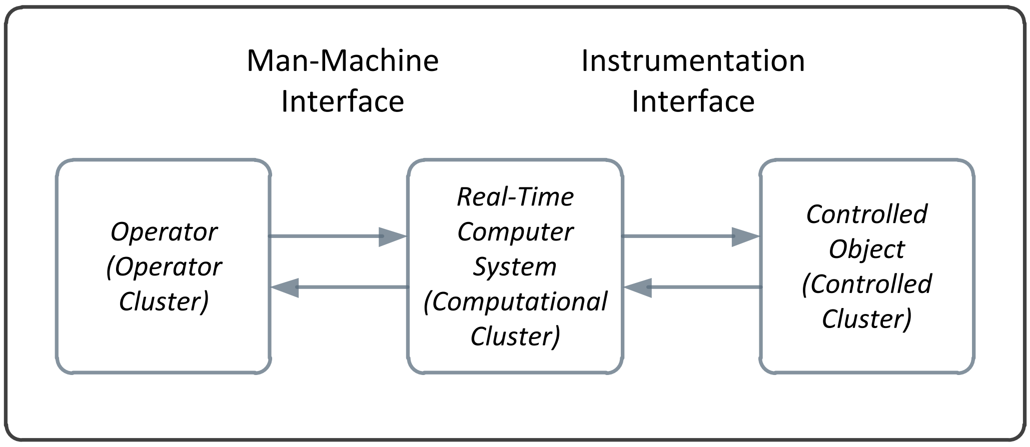

Man-Machine Interface
Instrumentation Interface
Operator (Operator Cluster)
Real-Time Computer System (Computational Cluster)
Controlled Object (Controlled Cluster)

Figure 3.1: Real-Time System [1]

input and output. The input user interface serves a human user for the entering commands which are than transferred by the real-time computer to the control object. The output user interface provides information about the controlled object to the human user. The **instrumentation interface** serves for the communication of the real-time computer with the controlled object. Its main tasks are:

- a translation of the physical signals acquired from the sensors into data which can be processed by the real time computer, and
- a generation of physical signals for the control environment which use them to control actuators.

A real-time system is not limited to a single set of these components, and depending on the application they can be implemented in groups called clusters.

The execution of the tasks in a real-time computer systems is constrained by deadlines. If the execution of a certain task must be done inside a given deadline for the results of the task to be valid the deadline is called firm. If a deadline is set for a task in a critical system and its violation causes serious damage to the system the deadline is called hard. In most cases the violation of a hard deadline causes catastrophic consequences and the complete failure of the system. In the case of a firm deadline, a system continues to operate normally and it only discards the results after the deadline is violated. If the violation of the deadline does not effect the system in any way and it can be tolerated such deadline is called soft deadline.Based on this characterisation of deadlines real-time computer systems are divided to hard (safety-critical) real-time and soft real-time computer systems.

As mentioned before, besides functional requirements, a real-time system must also meet temporal requirements of the application. The functional requirements are described as basic functions of the real-time system computer. The temporal requirements originate from the systems where the response of the system is conditioned by the time. Real-time systems are used in a wide range of applications and with respect to the type of the application they must possess a adequate level of quality. The quality standards are imposed by the dependability properties which can be classified into following four groups:

- **Reliability** is defined as probability that a system will provide a service for a certain period in time, given constant failure rates.
- **Safety** can be interpreted as a probability of a catastrophic failure of the system or the probability that the system will continue to provide service after a benign failure.
- **Maintainability** is the time needed for system to be repaired after a benign failure.
- **Availability** is a measure of the delivery of correct services with respect to the alternation of correct and incorrect service.
- **Security** is the ability of the system to prevent unauthorized access to information or services.

Real-time systems are modelled with the set of the significant state variables, these variables are called real-time (RT) entities. Each of these variables belongs to a certain subsystem and only this subsystem can change the value of the RT entity [1]. The RT entity is described by its attributes which can be divided in static attributes, that does not change with the time (i.e. name, type) and dynamic attributes that change with the time. A change of the value of the RT entity is captured with observations represented as an atomic data structure composed out of the name of the RT entity, the value of the RT entity and the time of observation. The observation is valid only for t a limited interval, therefore it can be described as temporally accurate picture of RT entity in time instant t and we call it real-time image [2].

3.1.2 Distributed approach

When a single application becomes too complex to be efficiently realized on a single centralized system, the work load is divided among several systems, thus forming a system of systems. To provide efficient functionality, the systems are interconnected and through exchange of messages they stay coherent. If these systems are real-time systems and they are interconnected with a real-time communication system, they form a higher level system, which is called a distributed real-time system.

A single node of the distributed real-time system is composed out of three integral hardware components:

- **Host computer** provides the computational functionality of the node.
- **Communication Network Interface (CNI)** ensures that operations of the communication network are seen as generic functions observed from the side of the Host Computer, with no concern on how they operate.

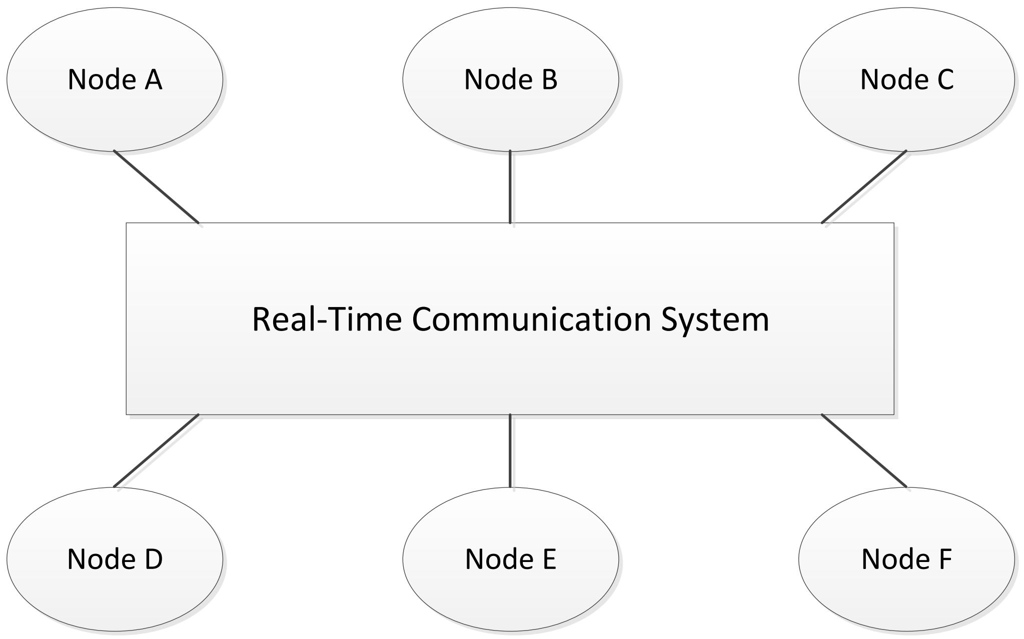


Figure 3.2: Distributed Real-Time System [1]

- **Communication Controller** provides physical connection to the communication network.

The advantages of the distributed real-time systems in comparison to the centralized solution stem from their ability to satisfy composability, scalability and dependability requirements.

Composability ensures that the independently created subsystems continue to provide full functionality after integration into a larger system. All properties foreseen for the single subsystems must be retained, independent from the configuration of the larger system. Each subsystem should be replaceable so that a subsystem with a different internal implementation but with the same functionality within the larger system can be installed instead.

Scalability of the system is determined via the ability to extend the system fundamentally while the complexity of the system stays within controllable boundaries. It requires that there are no limit on the extendibility of the system. The realization of the system in the form of a distributed real-time system provides outstanding extensibility, with acceptable increase in the complexity of the system.

Dependability of a distributed system depends on the ability of the system to achieve fault containment, error containment and fault tolerance. If a subsystem fails the larger system will continue to operate correctly or to achieve some safe state without any catastrophic consequences.The occurrence of a fault or error must be contained on subsystem level, this enables

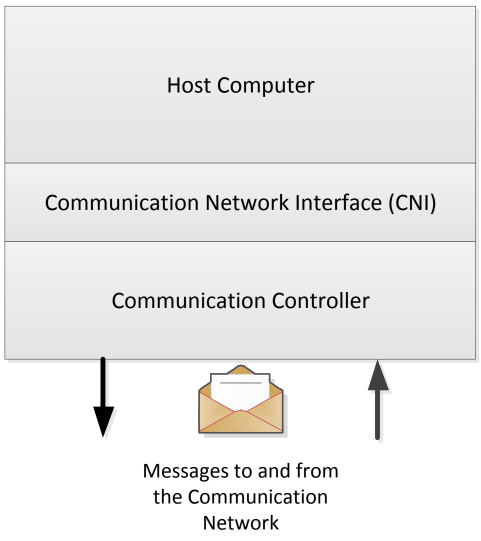

Host Computer
Communication Network Interface (CNI)
Communication Controller
Messages to and from the Communication Network

Figure 3.3: A node of the real-time system. [1]

designers to implement methods like a triple modular redundancy (TMR) to achieve the fault tolerance.

The Communication System

The other integral part of the distributed real-time system is the real-time communication system. It provides proper exchange of the messages among nodes of the system. If the transmission and reception of the messages are actuated by an event, we say that the communication system is **event-triggered**. The time instant of the message transmission is set by the host, these messages are than transmitted to the receiver and stored in the reception buffer where they will reside until the read event is initiated by the host of the receiver. Therefore, the timing behaviour of the communication is not regulated by and can not be influenced by the communication system. This type of the communication is not preferable for a distributed real-time system, because it fails to provide a timeliness under certain conditions, like faulty senders that send more messages then specified.

If the transport of the messages is regulated via a predefined time schedule the communication system is called **time-triggered** (TT). The host writes messages in to the memory of the CNI, these messages are then transferred by the communication system to the receiver at a predefined moment in time and also will be made available to the receiver host in the CNI memory at the known time instant. This allows hosts to communicate confidently with respect to the timing behaviour of the system. In the next Section 3.2 the basic theory about time triggered communication systems will be discussed.

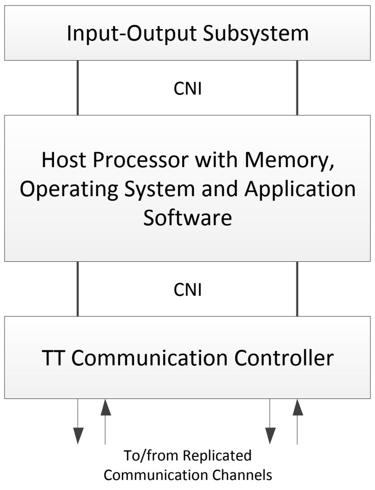

Input-Output Subsystem
CNI
Host Processor with Memory, Operating System and Application Software
CNI
TT Communication Controller
To/from Replicated Communication Channels

Figure 3.4: A node of the time-triggered architecture. [2]

3.2 The Time-Triggered Architecture

Every system is designed with the intend to carry out certain task. The requirements of the task and its complexity determine the design of the system. If the requirements of the application includes high composability, scalability and dependability the usage of the time-triggered approach would be a logical choice. The framework which enables such design is the Time-Triggered Architecture (TTA). It allows complex applications to be decomposed into clusters and nodes including the implementation of a system wide fault-tolerant global time.

Because the TTA is a framework for designing large distributed real-time systems the basic building block is a node.The internal structure of the node consists out of three parts: Input-Output Subsystem, Host Processor with Memory, Operating System with Application Software, Time-triggered Communication Controller (Figure 3.4). Several nodes connected over the TT communication system form a cluster (Figure 3.5). Special getaway nodes allow clusters to be connected with one another or with some external network. This reduces the complexity of the system and increases its dependability [2].

The communication system operates independently from the nodes and the messages are transmitted via a priori specified time-division multiple access (TDMA) schedule. The hosts store a message data in the CNI memory and at the given time instant this message is delivered via communication system to the CNI memory of the other node. The actions of the host computer can be planed a priori synchronous to the TDMA schedule [2].

3.2.1 The Global Time

The consistent perception of time among all nodes, is one of the most important requirements of the distributed real-time system. The actions of the system are executed on different nodes

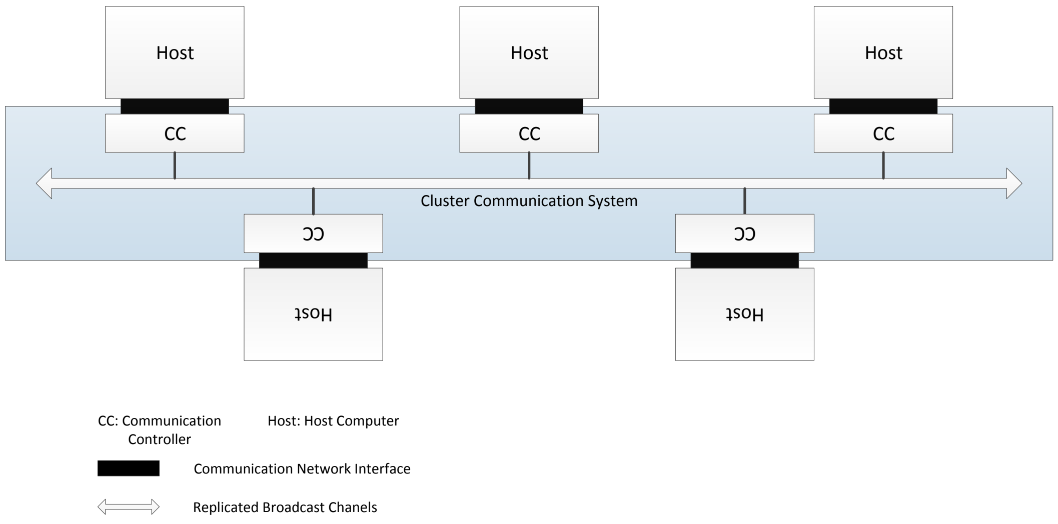

Host
Host
Host
CC
CC
CC
Cluster Communication System
CC
CC
Host
Host
CC: Communication Controller
Host: Host Computer
Communication Network Interface
Replicated Broadcast Chanels

Figure 3.5: Time-Triggered Architecture Cluster [2]

and it is of the essential importance that the temporal ordering of these events is guaranteed. Each node has its own local clock periodically driven by the quartz crystal oscillator, which generates an interrupt called **microtick** on each cycle. The duration between two microticks is called **granularity** g of the clock.

Because of the fact that each oscillator has slightly different frequency, the clocks driven by this oscillators will generate ticks at different rate. This is why it is impossible to perfectly synchronize two local clocks. The difference between two clocks is called **offset**. The maximal offset between two clocks in the system is called **precision** of the system Π.

The TTA introduces the notion of a global time to define weaker boundaries for the synchronization problem. The tick of the global time (macrotick) can be defined as a set of the microticks of the local clocks, under condition that the offset between two clocks is smaller than the precision of the system Π. We say that the global time t fulfils the **reasonableness condition** if

$$g < \Pi \tag{3.1}$$

holds for all local implementations of the global time. This condition ensures that the synchronization error in the system is never larger than one macrotick (Figure 3.6) [1]. After the global time is established the temporal order of events that occur on different nodes must be determined. In order to do this the notion of π/Δ**-precedence** must be defined. If two events occur on the same macrotick it means that they occurred within certain time interval π. Until the next occurrence of the event there is a time interval Δ (Figure 3.7). We say that a set of events occurring on the single macrotick is a π/Δ**-precedent** if the following condition holds:

$$[|z(e_i) - z(e_j)| \leq \pi] \vee [|z(e_i) - z(e_j)| > \Delta] \tag{3.2}$$

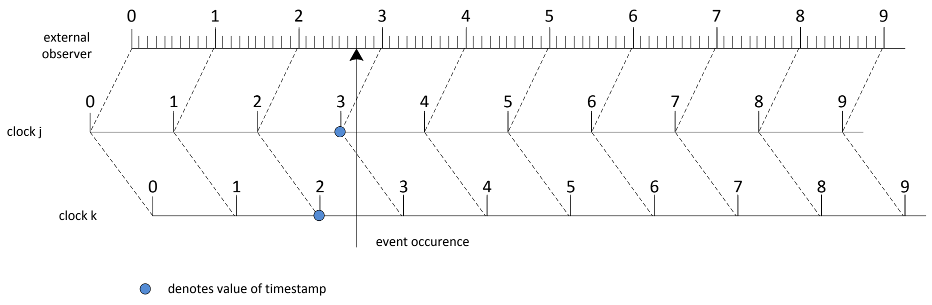


Figure 3.6: A timescale interpretation of the event occurance on two clocks [1].

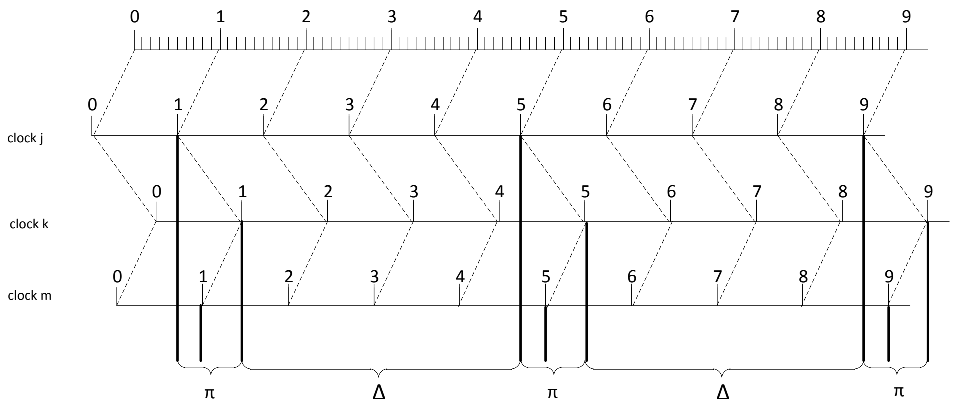


Figure 3.7: π/Δ-precedence depicted on three different clocks. [1]

The **π/Δ-precedence** states that events of one subset within the interval π are separated from the other subset for at least interval Δ.If the π is 0 than all events occur on the same time instant or they are a time interval Δ apart. At least $0/3g$-precedent event set is required to establish temporal order from the timestamps generated by the global time [1]. The time is usually represented with a directed linear timeline with infinite number of instants between any two occurring events. There are two representations of time [1]:

Assume a set $\{E\}$ of significant events which are of importance in a particular context. This set can be a tick of all clocks or the events of sending and receiving messages.If these events are allowed to occur at any instant of the timeline, we call the time base ***dense****. If the occurrence of the events is restricted to some sections of*

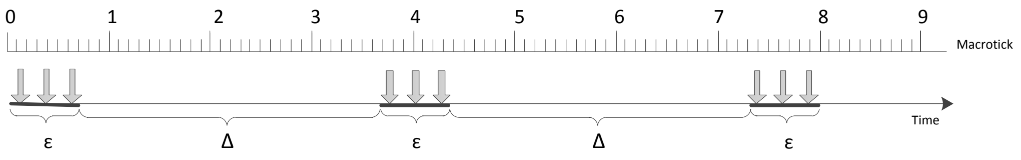


Figure 3.8: A graphical interpretation of the sparse time-base. Where the ε marks periods of the activity and Δ periods of silence [1]

.

the timeline, we call the timebase sparse.

The time-triggered architecture use time base where the events can occur only on certain instants, located within so called active intervals of time ε. Every two active intervals of time are separated with the silent interval Δ. The time base divided into active and silent intervals of time is called ε/Δ**-sparse** or simply **sparse** (Figure 3.8) [1].

3.2.2 Clock Synchronization

At the beginning of the previous Section 3.2.1 the synchronization problem was introduced. It is mentioned that due to imperfectness of quartz crystal oscillators two clocks on different nodes will generate ticks at the different rate. Also to implement a system wide global time the precision Π of the system must be smaller than the granularity of the global time. To ensure this all clocks in the system must be periodically resynchronized, the period between two resynchronization instants is called resynchronization interval R_{int}. The synchronization of the clocks within a cluster is called **internal clock synchronization** and it is independent to any external time line. After each resynchronization interval all clocks are adjusted to achieve sufficient precision. After the synchronization is preformed the clocks will drift again until next resynchronization interval. A synchronization condition of the ensemble of clocks is given by:

$$\Phi + \Gamma \leq \Pi \tag{3.3}$$

where

- Φ is convergence function and it denotes offset of the values immediately after resynchronization.

- Γ denotes drift offset, which is maximum divergence between any two good clocks from each other during the resynchronization interval R_int and can be calculated as

$$\Gamma = 2\rho R_{int} \tag{3.4}$$

  where the ρ denotes maximum specified drift rate.

The internal synchronization is carried out via synchronization algorithms.The simplest algorithms are so called master-slave synchronization algorithms. Which use a central master node to send its time value periodically to all other nodes.The Slave nodes compare their previously recorded time with the time from the slave and accordingly adjust their clocks.The synchronization condition for the systems with a central master algorithm is defined as

$$\Pi_{central} = \varepsilon + \Gamma \tag{3.5}$$

where ε denotes the latency jitter between the event of reading the clock value at the master and the events of message arrival at all slaves, which corresponds to the convergence function Φ [1]. The simplicity is the main advantage of this method, the drawback of the method is absence of fault tolerance.

Other way to synchronize the clock ensemble is with a distributed synchronization algorithm. These algorithms topically have three phases of the execution:

- In the first phase the time values from all nodes are exchanged among members of the network, such that every node has the status of the global time of every other node.
- In the second phase all values first examined for errors, then each node executes the convergence function to calculate correction term for the local global time counter. If the calculated correction term is larger then the specified precision of the ensemble the node is irreversibly out of the synchronization with the ensemble and it shuts itself.
- In the last phase each node adjusts its local clock to the calculated correction term.

The overview of the well known methods for clock synchronization in distributed systems can be found in the works of Tanenbaum [9] and Ramanathan [10]. The synchronization of the ensemble with the externally provided reference time is called external clock synchronization. The external source of execution is topically special computer with access to the high precision clock, like an atomic clock. The global time is represented in a format based on physical second, these formats are described in the follwing text of this thesis (see Section 3.2.3).

3.2.3 Time Standards

The measurement of the time can be based on astronomical events or atomic clock. The astronomical measurement of the time is done by a mechanical clock and it depends on the rotation of the earth. The rotation of the earth and its orbiting around the sun are not constant therefore the measurement is for certain applications not precise enough (e.g. critical real-time system). In order to provide more reliable time measurement the atomic clock was invented. A brief survey of the atomic clock story can be found in [11]. A atomic clock is generating time by counting the transitions of cesium 133 atom. The 9,192,631,770 transitions of the cesium 133 atom represent one second.

The world time based on atomic clock is produced by the Bureau International de l'Heure (BIH) in Paris, France. They collect time values from the atomic clocks of the international timekeeping institutes and by averaging them they produce International Atomic Time (TAI). This time is basis for all other time formats used today, the most important one is Coordinated

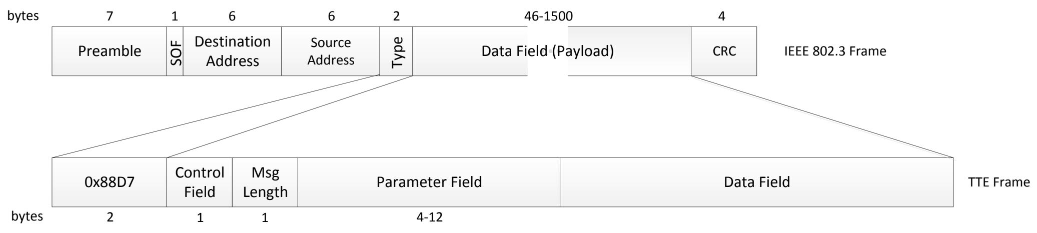


Figure 3.9: The format of the Time Triggered Ethernet Message. Upper block represents the IEEE 802.3 messages and lower block shows the TTE specific fields.

Universal Time (UTC). The UTC is a global standard for representation of time. Most of the clocks in the world are set by this standard. It is calculated from TAI by occasionally adding a leap second to compensate for Earth's slowing rotation [12]. The UTC has a discontinuous time scale and in order to calculate the correct interval between two timestamps the table of the leap second correspondences is required. Because of this many scientific applications that require such precise measurements use TAI representation directly. For the systems which have to be synchronized with standardized world time usually the UTC is used. Based on the TAI and UTC there are several other time formats like the Network Time Protocol (NTP) format [13],the format of the Global Positioning System (GPS) [14], the Uniform Time Format (UTF) [15],the Precision Time Protocol (PTP) format [16] developed for dedicated applications. The UTF is used by the Time-Triggered Ethernet protocol and it will be discussed in detail in the Section 3.3.

3.3 Time-Triggered Ethernet

Time Triggered Ethernet (TTE) is an implementation of the TTA (see Section 3.2) based on the IEEE standard Ethernet (more on Ethernet [17]).The TTE extends the standard Ethernet, which is non-deterministic, with the deterministic fault-tolerant time-triggered communication service, which provides a excellent infrastructure for the development of distributed real-time applications or distributed multimedia applications [18]. It can be used both for non-critical and safety critical applications.The TTE is designed to support both ordinary and safety-critical applications. The usage of safety-critical configuration requires more resources then the ordinary one. The TTE supports two classes of messages:

- Event-triggered (ET) messages
- Time-Triggered (TT) messages

The TT messages format is based on the format of the standard IEEE 802.3 messages (ET messages) (see Figure 3.9), with the specially designated value of the field *Type* for different types of messages based on the IEEE 802.3 standard. For the purpose of the TTE the frame type

$0x88d7$ is assigned by the IEEE EtherType Field Registration Authority [19].
The TTE system consist out of TTE Nodes, TTE Switches with the possibility to incorporate regular Ethernet nodes in the same network. The conflicts between ET and TT messages are resolved by preempting the ET messages so the TT messages can meet deadlines set by an a priori defined schedule. The schedule can be calculated offline or online depending on the requirements of the application.

3.3.1 Time Triggered Ethernet System

The TTE system is organized in clusters of TTE Nodes and any other systems with standard Ethernet, connected with the specially designed TTE Switch (Figure 3.10). The TTE have one master node called **rate master** which is used to disseminate global time to the rest of the cluster. The rest of the nodes are so called **slave** nodes, also called time-keeping nodes. In the fault-tolerant version of the TTE there is a node which serves as a secondary rate master master in case the primary rate master fails.The communication among members of the TTE system is established using the Time Triggered Ethernet Protocol. The **TTE Node** is a computer system composed out of four components:

- Host Computer,
- Hardware abstraction layer (HAL),
- Communication network interface (CNI),
- Time-Triggered Ethernet Controller.

The transmission of the ET messages is the same as in the standard Ethernet and if the Node does not require TT messages a standard Ethernet controller can be used. The communication of the TT messages requires additional resources and a specially designed infrastructure. The mechanisms used to provide time-triggered communication on top of the standard Ethernet can be implemented either in hardware or in software. These mechanisms are encapsulated in the subsystem of the TTE Node called TTE Controller. The **TTE Controller** is a multifunctional communication controller specially designed to be able to provide both the ET and the TT communication. In the CNI memory of the node a memory space is allocated for outgoing and incoming TT messages. On the reception of the TT message the memory place allocated for this message is overwritten with the data of the last received message. The host computer retrieves message data with a pull method. The transmission of the messages is regulated by the **TT dispatcher** so the conflict-free sending of the TT messages can be achieved. It reads message data from the CNI memory, written by the host with a push method, and transmits it on the next send instance. The ET messages are sent when the transmission of the TT messages is over.

Similar as the TTE Controller the TTE Switch also handles the TT and the ET messages differently. The ET messages are queued and transmitted accordingly when the communication channels are not used to transmit TT messages. If the ET message comes in to conflict with a TT message the TTE Switch clears the channel for the TT message.In such cases the ET messages are preempted and sent later. The maximum delay time of the TT message on the TTE Switch is called **switch transmission delay**, which is the time needed by the switch to clear communication channel currently used by the ET message.

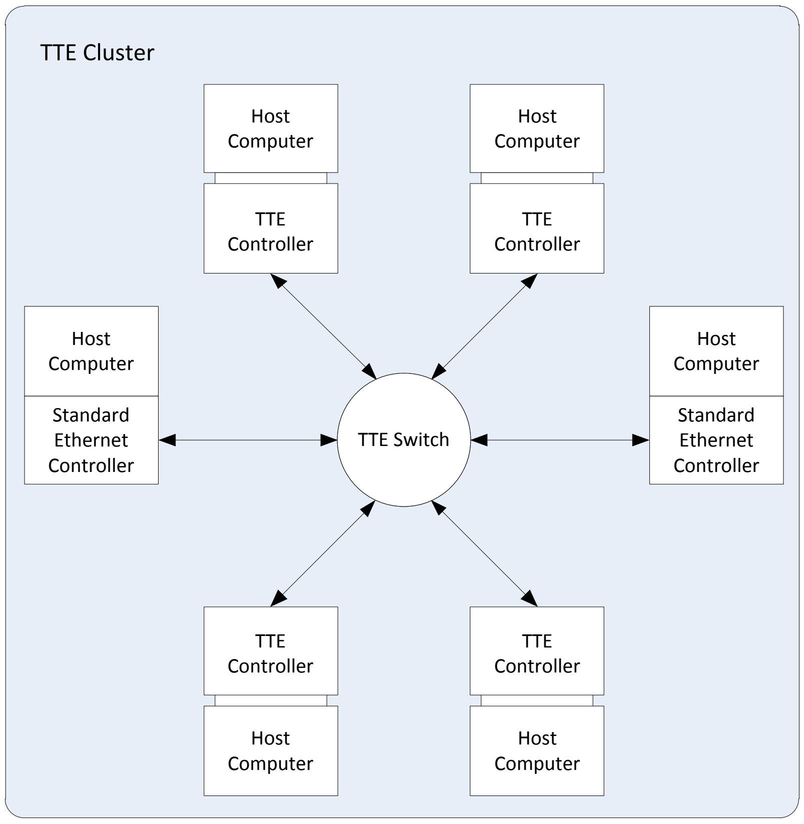


Figure 3.10: A typical structure of a TTE cluster [18]

3.3.2 TTE Services

The TTE is real-time communication system based on standard Ethernet and it provides fault-tolerant deterministic message exchange among the nodes of a distributed system. Without any additional hardware TTE is able to transmit and receive both ET and TT messages. These two functionalities together with several other can be arranged in six services to describe the complete scope of operation of the TTE:

1. Global Time Base and Service
2. Standard Ethernet Message Exchange Service
3. Time Triggered Message Exchange Service
4. Medium Access Service
5. Diagnostic Service

Global Time Base

The existence of the global time is of the great importance for the distributed systems where the temporal coordination of events in the system is needed. The concept of the global time and its most important aspects are introduced in the Section 3.2.1 of this thesis. The basic two goals of this service are generation of the global time and maintaining the certain level of synchronization among the members of the network. The global time of the TT Ethernet is represented in the UTF time format (Figure 3.11). This format is based on an 64-bit register where the upper 40 bits represent seconds and the lower 24-bits represent fractions of the second. The epoch of the UTF starts at the January 6, 1980 and it has an horizon of 34841 years.

The global time is disseminated by the rate-master node to the rest of the cluster via TT synchronization (*TTsync*) messages. The granularity of the global time is given in macrotics (MT), which is the basic unit of the global time. Every node in the cluster has a local clock which is used to generate macroticks on individual nodes. The correctness of the global time depends on the precision of the local clocks. The duration of the macrotic is configurable and its value should be equal for all nodes in the cluster, in order to maintain the consistent global time on all nodes. The granularity of the local clock is called microtick (μT) and it can be different for every node, because it depends on the hardware implementation and nominal frequency of the used oscillator. To preserve consistent length of the macrtotick every nodes adjusts the μT/MT ratio which is called **macrotick microtick conversion factor** (MMCF). The adjustment is preformed by a master-slave clock synchronization algorithm.

The global time of the cluster is based on the local time of the rate master node. The rate master node sends out the TT synchronization messages (TTsync) (see Section 3.3.3 to all other nodes. On the reception of the the TTsync message the synchronization data of the message is stored in the buffer accessible by the host computer. The host computer uses this data to calculate and apply clock correction term. The clock correction term is calculated as follows:

$$\Delta R_i = \tfrac{MTD_i}{TS_i - TS_{CS}}$$

The terms in the formula are defined as:

- TS_{CS} is the time stamp of the received TTsync message stored on reception,
- MTD_i is the measured time difference at the received synchronization frame i,
- TS_i the time stamp of the measurement and
- ΔR_i is the difference of rate between local node and rate master node.

The final rate correction term is calculated as the average value of two rate difference values:

$$RCT = \frac{\Delta R_1 + \Delta R_2}{2} \tag{3.6}$$

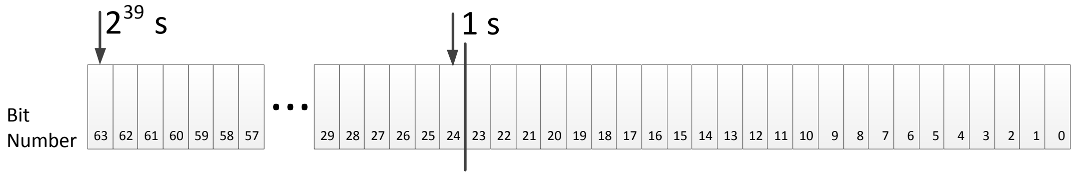


Figure 3.11: The time format of the TT Ethernet is implemented with the 64-bit register.Where the last forty bits are measuring seconds and the rest the second fractions.

The rate correction is realized on all nodes at the same time instant, this point in time has distinct period and phase and it is called **synchronization instant**. As it is stated above the rate correction is applied by changing MMCF:

$$MMCF_{new} = MMCF_{old} + RCT \tag{3.7}$$

If the difference of the global time on a local node is too large the slave node applies state correction instead of rate correction, by setting the value of the global time register on the slave node to a value of the rate master node.

The rate master node is able to synchronize its time to the external time (i.e. GPS), the rest of the system is then synchronized with the internal clock synchronization algorithm. The rate master node has no knowledge of the synchronization of the cluster, meaning the rate master node does not know whether any slave is synchronized.The TTE provides no protection against possible attacks on the global time. In the following chapters these problems are isolated and solutions are proposed to counteract them.

Standard Ethernet Message Exchange Service

The communication of ET messages is implemented as a store-and-forward paradigm with the best-effort delay specified by the Ethernet standard [18]. It requires no communication schedule and if a node requires no TT traffic a standard Ethernet controller can be used. More on standard Ethernet message exchange can be found in the official document of the IEEE 802.3 standard [17].

TT Message Exchange Service

The exchange of the TT messages is executed by the a priori defined schedule. In the schedule of the TTE 16 different **periods** (Figure 3.12) can be used. The TT messages can be configured to be transmitted in each of these 16 periods. All the 16 periods are a power of two of the base period, which is usually $1s$ but can be configured as needed. The periods can be represented on the global time counter such that the portion of the register left from a period bit is increased by one each time the cycle of that period passes. Within the periods the **offset** (Figure 3.12) can be designated for the TT message. The offset is represented with the 12 bits right of the period bit.

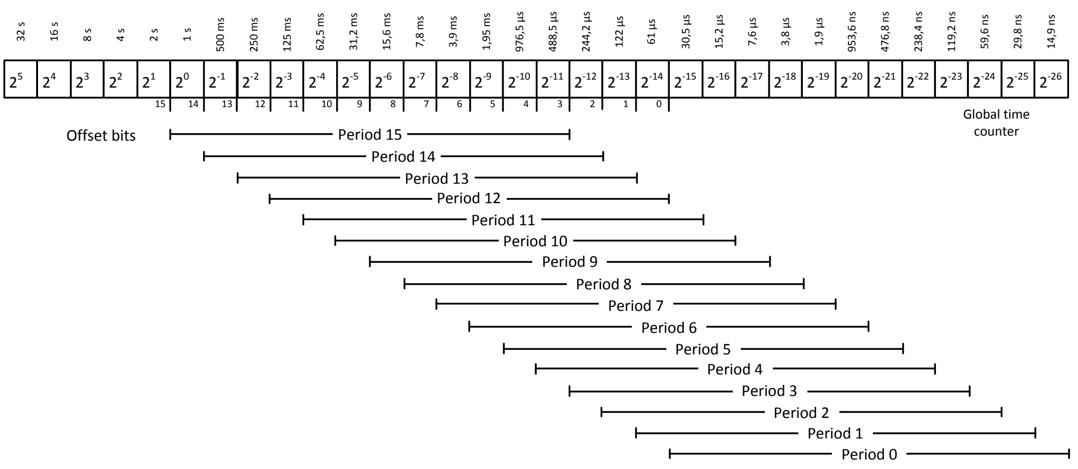


Figure 3.12: TT Ethernet periods and offsets for the base period of one second. [20]

The exchange of the TT messages depends on the type of the TT message. In the TT Ethernet two types of TT messages can be distinguished: **periodic** and **sporadic**. The periodic TT messages are usually used to transmit state information of the node, they are transmitted periodically in every cycle of the period for which is the message configured. These messages are transmitted from the CNI memory of the sender to the CNI memory of the receiver, where each message has a reserved place in both instances of the CNI memory. Upon the reception of a message the old message in the CNI memory of the receiver will be overwritten. The host can access the message by *pulling* the information from the CNI memory.Although they are not delivered periodically the resources for these messages are allocated a priori. Therefore they are much similar to the periodic messages. The sporadic TT messages are sent only if the host computer have updated the message since last transmission and the information is delivered in the push mode to the host computer. They are used to transmit sporadic events in the network.

Every TT message has a specified reception window, and only messages received within this time interval are considered to be valid. This time is dynamically recalculated after every received message for the next message. The reception window with configurable size is defined to compensate for the drift offset. The quality of the synchronization in the cluster is the main factor by choosing the size of the receive window.

There exists also a constant delay for TT messages which is caused by the network elements like switches and the length of the cables. This delay is also called propagation delay and it is stored in the configuration memory of the TT controller. Each node has the propagation delay for any other node in the network.The propagation delay can be changed on runtime if the configuration of the network changes.

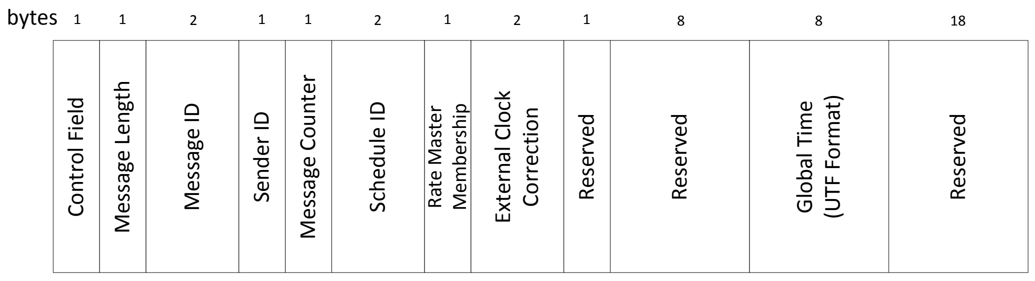


Figure 3.13: The format of the TT synchronization message. [20]

3.3.3 TT Message Types in TTE

As it is mentioned above, the TTE supports both ET and TT traffic, where the ET messages are handled by the IEEE 802.3 standard.The focus of this thesis is to describe how the TTE handles the TT traffic and how it can be improved with the spotlight on the security. The TT communication of the TTE is particularly designed to provide services needed for establishment of deterministic fault-tolerant communication. Four types of TT messages can be distinguished:

- TT Synchronization Messages (TTsync)
- TT Data Messages
  - Sporadic TT Messages (TTsM)
  - Periodic TT Messages (TTpM)
- TT Diagnostic Messages (TTdiag)

The Data Field of a TT message defined by the IEEE 802.3 standard contains a TT message header and the TT message payload. The first byte of the TT message header, which is 12 bytes long, is called Control Field.It contains the type of the TT message, the message update bit and the last message bit which marks the last message of the sequence.

TT Synchronization Messages

The synchronization messages are a special class of messages sent out by the **rate master** to the rest of the cluster to establish clock synchronization. The length of these messages is fixed (34 bytes) and they are identified by the value $0x74$ of the *Control Field* (Figure 3.13). They contain all information needed for the clock synchronization. The reception timestamps of these messages and the global time information in the message are used by clock synchronization algorithm to calculate clock correction term for a node. The reception timestamp of the synchronization messages is adjusted with the propagation delay to simplify the clock synchronization algorithm.

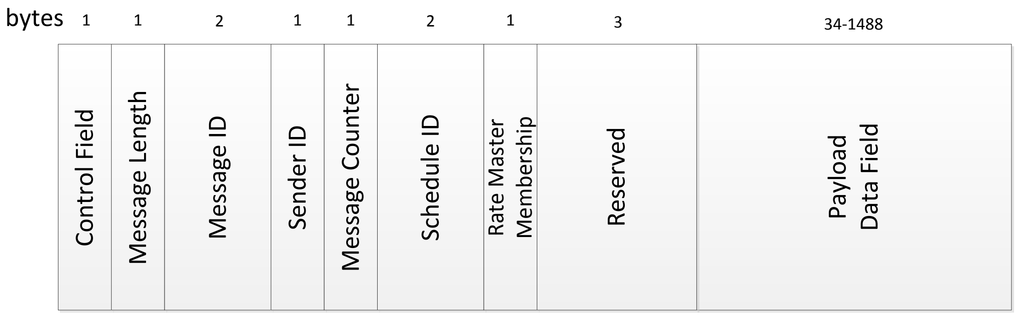


Figure 3.14: The format of the TT data message [20].

TT data Messages

The TT data messages (Figure 3.14) are used to transmit an application information among the nodes of the cluster. These messages are highly customizable and can be defined as periodic or sporadic. The length of these messages is limited to the maximal length of 1488 bytes. The TT data messages can be scheduled in any period. The period of a TT data message is encoded into the *period ID* field of the message header, which together with the offset of the message is used as a distinct identifier for the TT data message. **Periodic TT Messages (TTpM)** are sent repeatedly and each new message rewrites the old message in the CNI memory. The host access the TTpMs with the information pull mode and the sender receives no acknowledgement whether the message is received or not. The control field of the message has a three distinct values described in Table 3.1. The **TT sporadic message (TTsM)** is a type of a TT message

TTpM status	Control Field Value
not last message, not updated by host	$0x20$
not last message, updated by host	$0x24$
last message updated by host	$0x2C$

Table 3.1: TTpM Control Field

that is transmited only if the content of the message is updated by the host. It also has three different values of the control field Table 3.2

TTsM status	Control Field Value
not last message	$0x34$
last message	$0x3C$

Table 3.2: TTsM Control Field

TT Diagnostic Messages

The TT Diagnostic messages (TTdiag) are used to transmit local diagnostic data and the diagnostic data about the local view of the global state of the system . The header of the TTdiag messages corresponds the header of the TT periodic messages. The local diagnosis data is collected and updated by the TT Controller, but there is the possibility, to allocate space at the end of the TTdiag message if the host needs to transmit diagnosis data.

3.4 Security Concepts

The following section describes basic security concepts, techniques, and mechanisms used to protect assets of the computer systems against any threats against these assets. The overall usage of computer systems increased drastically in past few years and one of the main requirements of these systems is security (see Section 3.1). Because more and more computer systems use a connection to the internet or to other system in order to provide certain functionality, their assets become even more exposed to diverse threats and it is of essential importance to provide them with adequate protection. This becomes even more important if the computer system is safety-critical or it is part of a larger safety critical system. The same standards hold for distributed real-time systems and the security aspect is even more important than in computer systems in general. For example if in a modern car the brake-by-wire system is compromised in anyway it could lead to catastrophic consequences. **Computer security** is defined as [21]:

> *The protection afforded to an automated information system in order to attain applicable objectives preserving the integrity , availability, and confidentiality of information system resources (includes hardware, software, information/data, and telecommunication).*

Therefore, main objectives of the computer security are [21]:

- **Confidentiality** ensures that confidential information stay undisclosed to unauthorized subjects and that the privacy of the individuals which own this data is completely respected. Which means that only they can decide how and with whom they wish to share this information.
- **Integrity** assures that data can only be changed in supervised and authorized manner and that the system operates under optimal intended function without any accidental or intentional manipulation.
- **Availability** assures that system is not any way denied to intended users.

These three concepts represent the most general description of security which can also vary depending on the type of the system. Often an introduction of new concepts is needed in order to fully define security of the given system. The most common concepts additionally introduced to describe security of computer systems are authenticity and accountability. **Authenticity** assures that the subject can be verified as genuine and can be trusted, also the data received in transmission can be validated.Authenticity can be observed as a subset of integrity.**Accountability** assures that actions of an entity can always be traced to that exact entity. It comprises non-repudiation, deterrence, fault isolation, intrusion detection and detection, and after-action recovery and legal action [21].

Security represents dependability of a system with respect to prevention of unauthorized access and/or handling information and/or availability.For safety critical systems security is subset of safety and for such systems any unauthorized manipulation or access can result in accident. For systems where unauthorized access may result in accident, security is needed but not in the same extend as in former case [22].
The threats on assets of the computer system can be classified in three categories [21]:

- **Hardware threats** include any actions which would make hardware elements of the system indisposable in any way. This type of threats mainly bring the availability of the system at risk.

- **Software threats** comprise any attacks against the operating system, applications or utility drivers running on the system. Threats on software in dependence of the type of an attack can influence all three main security properties of the computer systems.

- **Data threats** consider unauthorized actions on data files of the systems. These actions can be destruction, manipulation, or duplication of data files.

- **Communication Channels** threats are divided in two type of attacks passive attacks and active attacks. **Passive attacks** represent eavesdropping or monitoring, where the attacker passively collect confidential information about the system. **Active attacks** appear during runtime and consist of modifying or replaying data, masquerade, or denial of the service on the communication channel.
  Modification of data means that a portion of the message is change with false data.The **replay** attack consist of retransmitting valid messages to create malicious effects. **Masquerade** means that a attacker pretends to be a valid user and as such tries to infiltrate the system. The **denial of service** represents any obstruction of a regular operation of the system.

For the protection of computer systems various security services and applications are developed. These services ensure that security requirements of the system are fulfilled. The important part of these services are cryptographic techniques and mechanisms which provide means in order to accomplish security objectives. **Cryptography** is defined as [23]:

> *The art and science of keeping messages secure, by applying a encryption algorithm, also called cipher, to the plaintext of the message in order to disguise it in to ciphertext.*

The basic cryptographic algorithms used to provide the security properties to the computer system can be summarized in five following categories:

- Symmetric Encryption
- Message Authentication

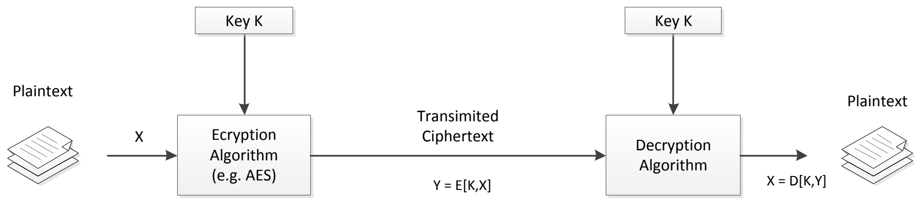


Figure 3.15: The plaintext is encrypted on the left side and sent over the network. The message contains ciphertext which is than decrypted on the right side by the decryption algorithm. [21]

- Public Key Encryption
- Digital Signature
- Key Distribution

3.4.1 Symmetric Encryption

The **symmetric encryption algorithms** disguise plaintext messages using a single key. The same key is used to decrypt the ciphertext by a decryption algorithm (see Figure 3.15).The main goal of symmetric encryption algorithms is to provide confidentiality and protect data from passive attacks.This way of encryption has been used since ancient times (e.g. Julius Caesar). The symmetric encryption algorithms can be classified in two major groups block ciphers and stream ciphers. The **block ciphers** encrypt chunks of plaintext (i.e. 64-bit or 128-bit), whereas the **stream ciphers** encrypt data one byte at the time. The symmetric encryption algorithm family has a large number of members, where the block ciphers are more common and the stream ciphers are more useful in particular applications. The most important symmetric encryption algorithms are the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES). Both algorithms are block ciphers with the block size of 64-bit by the former and 128-bit by the latter. The DES algorithm is considered to be one of the most resistant algorithms to cryptanalysis. Today a brute-force attack is considered as the only way to brake the DES algorithm. The increase in computational power of computers exposed somewhat the DES to brute-force attacks. Although it is still widely used the National Institute of Standards (NIST) of the United States published the new standard AES eventually to replace DES. The key length of the AES range from 128 to 256 bits and even in shortest case are sufficient to provide enough protection for the foreseen future.

3.4.2 Message Authentication

Even if the received message is encrypted with the valid key, this does not prove that it is genuine. The encrypted message could be changed or falsified with an active attack. In order to

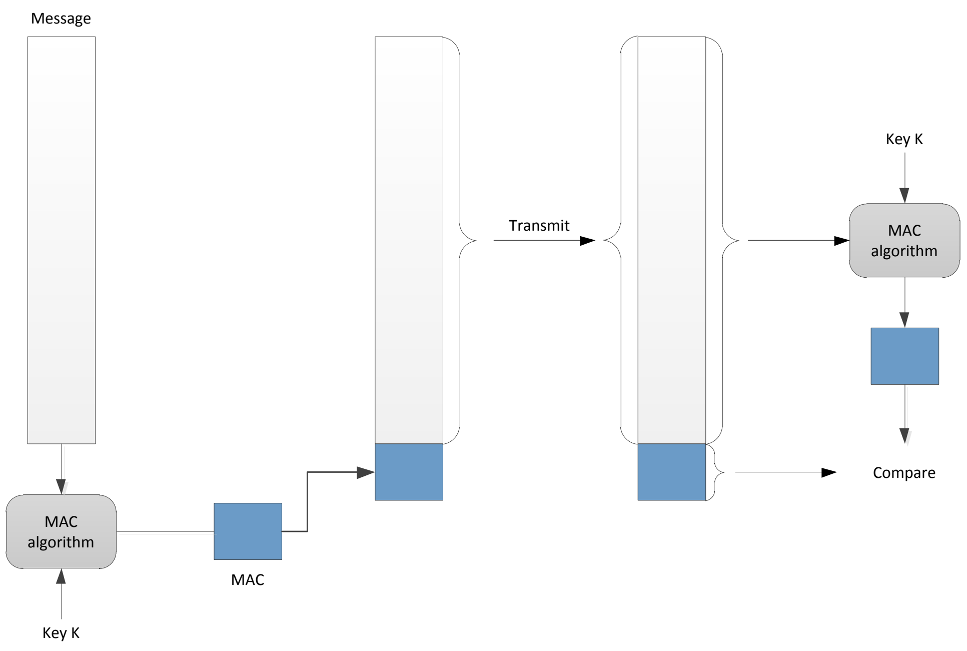


Figure 3.16: The creation, transmission, and verification of the Message Authentication Code [21]. The light gray block is the message and the blue block represents the MAC.

prevent active, attacks message authentication algorithms must be implemented. Most common techniques for message authentication are: Message Authentication Code, One-Way Hash Function, and Secure Hash Function. In the scope this thesis the Message Authentication Code (MAC) method is used, which is described in more detail in following text. For the prove of concept either method could be used, the MAC is chosen because of the more convenient implementation.

The Message Authentication Code (MAC)

From the plaintext of the message using a encryption algorithm a small block of data is generated. This block of data is also called Message Authentication Code or Message Tag (see Figure 3.16). The encryption algorithm used to generate the MAC can be any symmetric algorithm, therefore it also uses symmetric keys. The encryption in this case must not be reversible because to authenticate a message a receiver also runs the message trough the same algorithm. If the both tags match the message is said to be authentic. The plaintext is encrypted with a symmetric algorithm and the MAC is created out of the ciphertext (e.g. CMAC, OMAC). The MAC can also be generated from a hash function (e.g. HMAC) with the similar principle as in the former case.

3.4.3 Public-Key Encryption (PKE)

The Public-key Encryption is based on the concept of the **asymmetric** keys. It is first introduced by Diffie and Hellman in 1976 as new direction in cryptography [24]. The basic principle of Public-key cryptography is that instead of a single key every entity has two keys,a **public key** and a **private key**. In communication of two entities A und B, first they need to exchange the public keys, while the private keys remain secret. If A wants to send an encrypted message to B it encrypts message with the public key of B. The entity B decrypts message with its private key. Depending on the application PKE can be used to ensure confidentiality or authentication and data integrity. The applications of the PKE can be classified in three groups: symmetric key distribution, digital signature, and encryption of secret keys.
The most commonly used PKE technique is called RSA by it authors R.L.Rivest, A.Shamir, L.Adleman, published in 1977. Since than it became one of the most implemented techniques in cryptography (e.g. Secure Shell (SSH) [25], Transport Layer Security (TLS) [26]).In the last couple of years a new scheme called Elliptic Curve Cryptography (ECC) has gained on popularity and it is considered to be concurrent to RSA. It was first proposed by two authors independently in 1986, Miller [27] and Koblitz [28]. The ECC provides the same level of security as RSA with shorter keys which brings a significant increase in performance as the length of the key increases [29]. The RSA is on the other side more analysed and it has greater confidence level.

3.4.4 Digital Signature

For applications which require authentication and non-repudiation at the same time the digital signatures are certainly most effective solution. The process of digital signature starts with creating a hash from an plaintext message with a hash function. The hash is than encrypted using its private key to create a signature. The message is than sent together with the signature. The receiver first calculates the hash of the message with the same hash function. Than the receiver decrypts the signature with the public key of the sender and compares the calculated hash with the decrypted hash. If the hashes match, the receiver is certain that the message is authentic and that it comes from the right sender. The most common algorithm for the generation of digital signatures is the Digital Signature Algorithm (DSA) as proposed by NIST in the Digital Signature Standard (DSS) [30] [31]. For the purposes of digital signatures in scope of this thesis the Elliptic Curve Digital Signature Algorithm (ECDSA) [31] has been used. It is a version of DSA which uses ECC for public key encryption.

3.4.5 Key Distribution

The secure distribution of a key is equally important as the encryption of the message itself. The exchange of public keys is done with so called **public key certificates**, which allow the key exchange with the prove of authenticity. The most common standard used for representation of public key certificates is X.509 which is applied in numerous network protocols and applications (e.g. SSH [25], TLS [26]). For the exchange of symmetric keys a PKE schemes such as Diffie-Hellman key exchange [24] can be used.

CHAPTER 4

System Model

The introduction of the x-by wire technology [1] in transportation systems increased the number of distributed real-time computer systems in cars, aircraft, trains, ships, ect. These are only few examples where safety is a crucial property of the system and the lack of security endangers safety. The importance of security is not only bounded on such safety critical systems with high safety concerns. The lack of the security endangers also other systems (e.g industrial plants) and their valuable assets. Let us consider for example a control of several valves in an industrial plant, where each valve is controlled by a real-time network node. Each valve controls a flow of the certain chemical in to the industrial chemical process. The actions executed on valves must be perfectly synchronized throughout the real-time network.If even a slight deviation occurs in the operation of the distributed real-time system, which controls these valves. It could lead to serious problems and even catastrophic consequences (e.g. the chemicals are wrongly mixed and the reaction starts fire). It is therefore of essential importance that the synchronization algorithm can not be compromised in any way. This means that the global time is protected from accidental and malicious faults.

If an attacker could gain access to the system and execute a malicious attack against the global time on one of the real-time network nodes, such that the global time on the node drifts from the actual global time, which effects could arise? The answer to this question is: unaligned control of the valve on the faulty node with the rest of the system. One of the consequences for the industrial process is late or to early application of the chemical controlled by this node. This could ruin the whole proces, create dangerous chemical reaction (i.e. fire, radiation). If the attack is executed with such precision and patience that the deviation is impossible to discover, with the time the deviation can reach a critical point and cause unrepairable damage. Eventually, the fault is discovered but the damage is done and valuable resources must be spent on fixing the problem. In the worst case the consequences of such a short time drift can be a failure of the industrial plant or even loss of a human life. If the threat is discovered in time, the negative consequences can be avoided.

In this chapter the basic system model for the protection of the global time base in TT systems is described with the possible threats, requirements on security and dependability, and

[1] x stands for safety related application such as steering, breaking, suspension control, powertrain etc.

methods to provide them. In the example described above the sensibility of the security issues in general have been established. Now we want to go further and pinpoint the discussion on TT systems. The focus will be on communication channel threats and ways to counteract them.In particular we will discuss active attacks against the global time and augmentation of the current clock synchronization algorithm of the TTE, as the implementation platform, with the set of security mechanism to provide protection against malicious attacks. The security layer for the TT systems provides authenticity and non-repudiation to the global time.

4.1 Security Threats and Requirements

A malicious attack against TTE system can be directed towards the host or towards the communication channels of the system. The hosts are usually black box systems and it is very hard for the unauthorized user to get access to the system. But very often these systems are designed with security weak spots (i.e. buffer overflow) that can be exploited to carry out the attack. If the unauthorized user gets access to the system, he can change or deny its functionality.

If the host of TTE node is considered secure, the most serious threats against the TTE system are attacks against its communication channels.The basic functionalities in danger are the clock synchronization and the global time base. The precise notion of the global time is the basic principle in the real-time systems and any unauthorized access to the global time puts the whole system in imminent danger of failing catastrophically. Connections among nodes of the TTE system are external and can be easily accessed. Therefore the TTE systems are also vulnerable to both passive and active attacks. The difference between standard Ethernet and the TTE is the timing of the messages must also be considered. In the Chapter 3 the types of messages in TTE

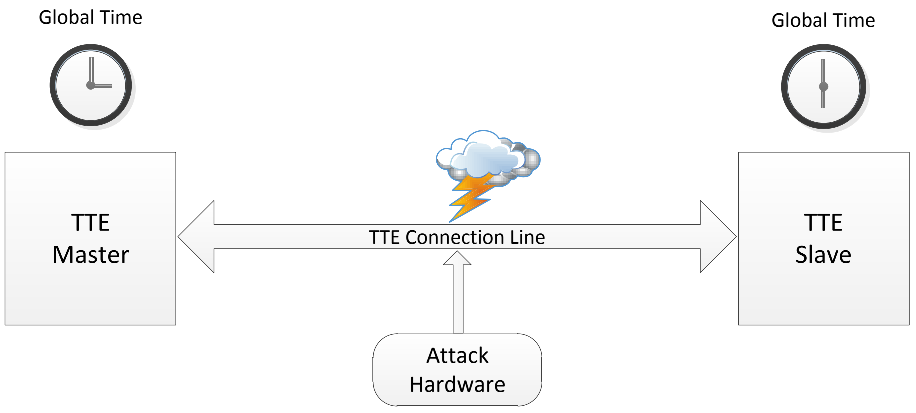


Figure 4.1: The time drift communication channel attack scenario. With the attacker connected on the communication line between a master and a slave altering the global time.

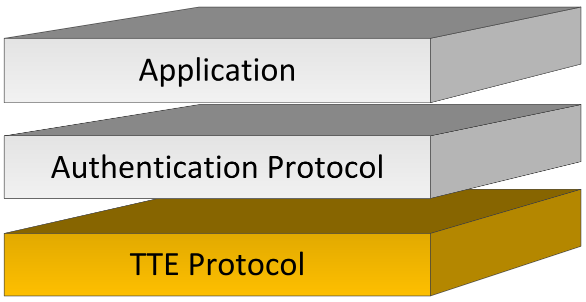


Figure 4.2: The Layer perspective of TTE with secure clock synchronization algorithm. First block from the bottom represents the TTE system functions, second block is divided in two parts a application and the secure clock synchronization protocol.

have been described. The application running on the host is responsible for the data messages security. The information carried by these messages can be protected by conventional cryptographic techniques described in Section 3.4. The TTE protocol ensures the timing behaviour of data massages. The security of the clock synchronization algorithm depends on the security of synchronization messages used for dissemination of the global time. With the special hardware, connected to the external lines of a TTE network, the delivery time of any TT message can be delayed or pushed ahead by an attacker. The reception of TT messages is expected in the reception window for the message. If the message is received within this window, it is considered valid, if not, the TTE controller will discard it. This way of causing damage can be considered as active attack, the combination of modification and denial of service attacks.

The manipulation of the synchronization messages by the attacker causes the global time to drift from the actual value and it creates a false view of time at the targeted node (see Figure 4.1).This kind of attack against synchronization messages affects the whole system, because although only one node has a false sense of time, the TT communication with other nodes and the execution of the tasks are not aligned with the rest of the system. TTE provides no protection against such attacks. A special security mechanism is needed, on top of the existing clock synchronization algorithm to neutralize this threat. The main security requirement in this case is authenticity of the global time. The global time must be authenticated for every node by the trusted authority and the result of this process has to be delivered to the individual nodes also in an authenticated manner. This will make the information about the state of global time on each node available system-wide. In the following section this mechanism is described in more detail.

4.2 The Secure Clock Synchronization Algorithm

The current clock synchronization algorithm uses synchronization messages to disseminate global time information for a cluster of TTE nodes. TTE is designed as a fault-tolerating protocol and

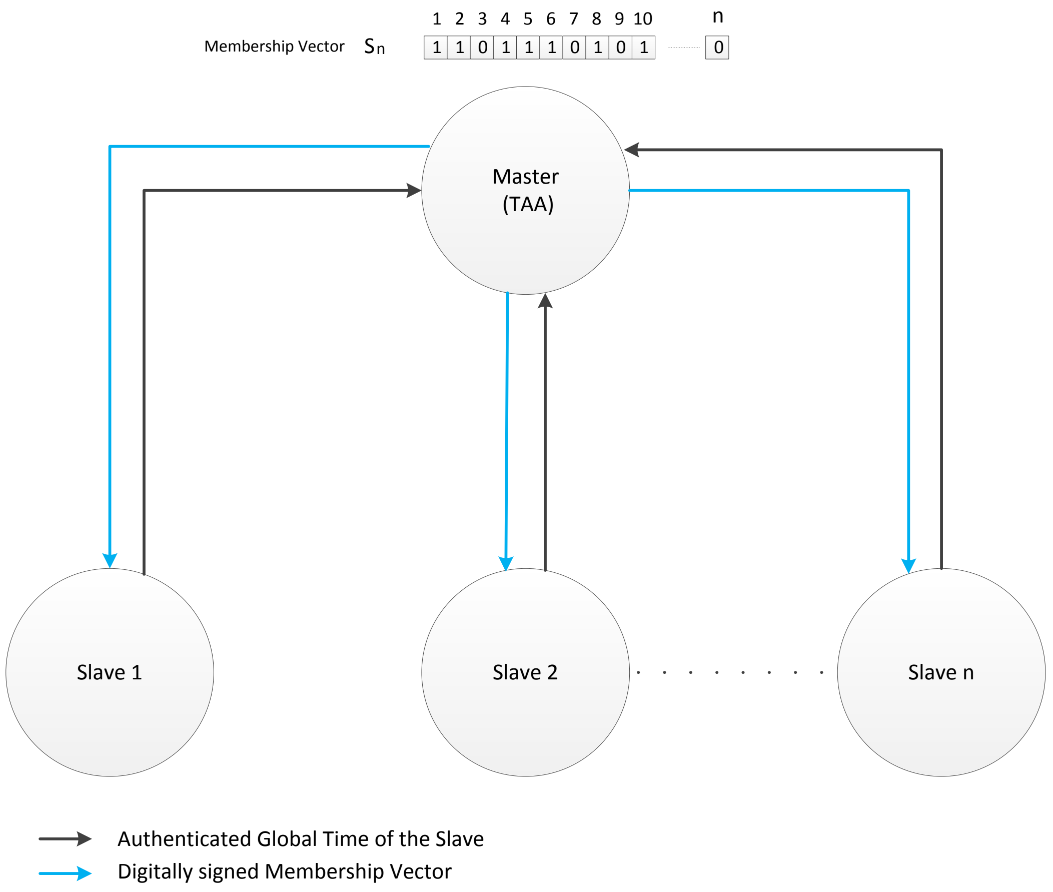


Figure 4.3: The Secure clock synchronization protocol with three nodes and the master. The two line colors represent two types of the messages used by the protocol. The secure membership vector is stored on the master and only the master can change its content.

it provides certain level of protection against accidental faults but none for intentional malicious faults. The introduction of cryptographic techniques would require redesign of the whole protocol. The authentication protocol is designed as a security layer on top of the current clock synchronization algorithm because the secure clock synchronization is an optional service. The authentication protocol is also called a secure clock synchronization protocol. The implementation of the authentication protocol requires no changes in the TTE protocol or the synchronization protocol and it can be considered as a security abstraction layer for the TTE. In the Figure 4.3 the structure of the TTE System with the secure clock synchronization protocol is depicted. The secure clock synchronization protocol operates in the background of the application using the standard TTE protocol.

The authentication protocol uses the master/slave principle of the communication. Beside being the global time authority, the master node is also designated to be a trusted authentication authority (TAA). The master node receives the global time from each slave in authenticated messages. If the authentication code of the message is genuine and the value of the slaves global time corresponds to the actual global time of the master, the slave is marked as **trusted**. If the security or global time checks fail, the slave is marked as **untrusted**. The security status of the slaves is saved in a **membership vector**, so that value 0 is designed to the untrusted node and value 1 to the trusted node. The membership vector is transmitted so that each slave has access to it and can act upon the information it contains. The application on the host of the slave node decides how to handle nodes with a certain security status. The messages of the authentication protocol must be protected against malicious attacks. The messages sent from the slave to master must be authenticated using message authentication techniques described in the Section 3.4. The authenticity of the message is required because the master must be able to verify that certain value of the global time comes from the genuine slave. The messages containing membership vector sent from the master, must be digitally signed so that the authenticity and non-repudiation can be verified.The non-repudiation is required if the results of the protocol have to be elaborated to the third party. This is useful in cases of legal actions in the case of failure of a system, where the system must be able to show why a certain action has been executed. If the slave node executed actions as the result of the protocol status change it must be able to show who is accountable for the cause of that action and be able to trace the source of the problem. This is important because only the master should be able to create the membership vector. The usage of symmetric message authentication algorithms in former and digital signatures in latter case ensures required security properties.

The secure clock synchronization protocol can only be activated, after all nodes in the network have been started and the synchronization among these nodes has been established.The global time of the TTE system is established after at least one synchronization message reaches all nodes. The authentication protocol is executed periodically. It transmits messages according to a predefined schedule. The secure clock synchronization requires two types of messages, the membership vector message transmitted from the master to the slaves and the global time message transmitted from slaves to the master. The schedule for the authentication protocol is part of the TTE protocol schedule. The resources for the authentication protocol must be reserved a priori and it must consider the execution times for the cryptographic algorithms. The order of the authentication protocol messages in the schedule depends on the implementation.

The execution of the protocol is implemented in several stages. In the first stage, so called initialisation stage of the protocol, the master transmits the membership vector to the slaves with no trusted members, as they are jet to be verified. The values of the vector in the initial round are ignored by the application. In the same round all slaves send their global time values in an authenticated message to the master. After global time messages from the slaves are delivered, the master verifies the authenticity of the messages and only if the message can be authenticated, the global time value contained within the message is compared to the masters local global time value. If the global time value received from the master corresponds to the actual global time, the slave node is marked as trusted and value 1 is written in the membership vector cell for that slave. In the next round the membership vector, with updated values, is transmitted to the

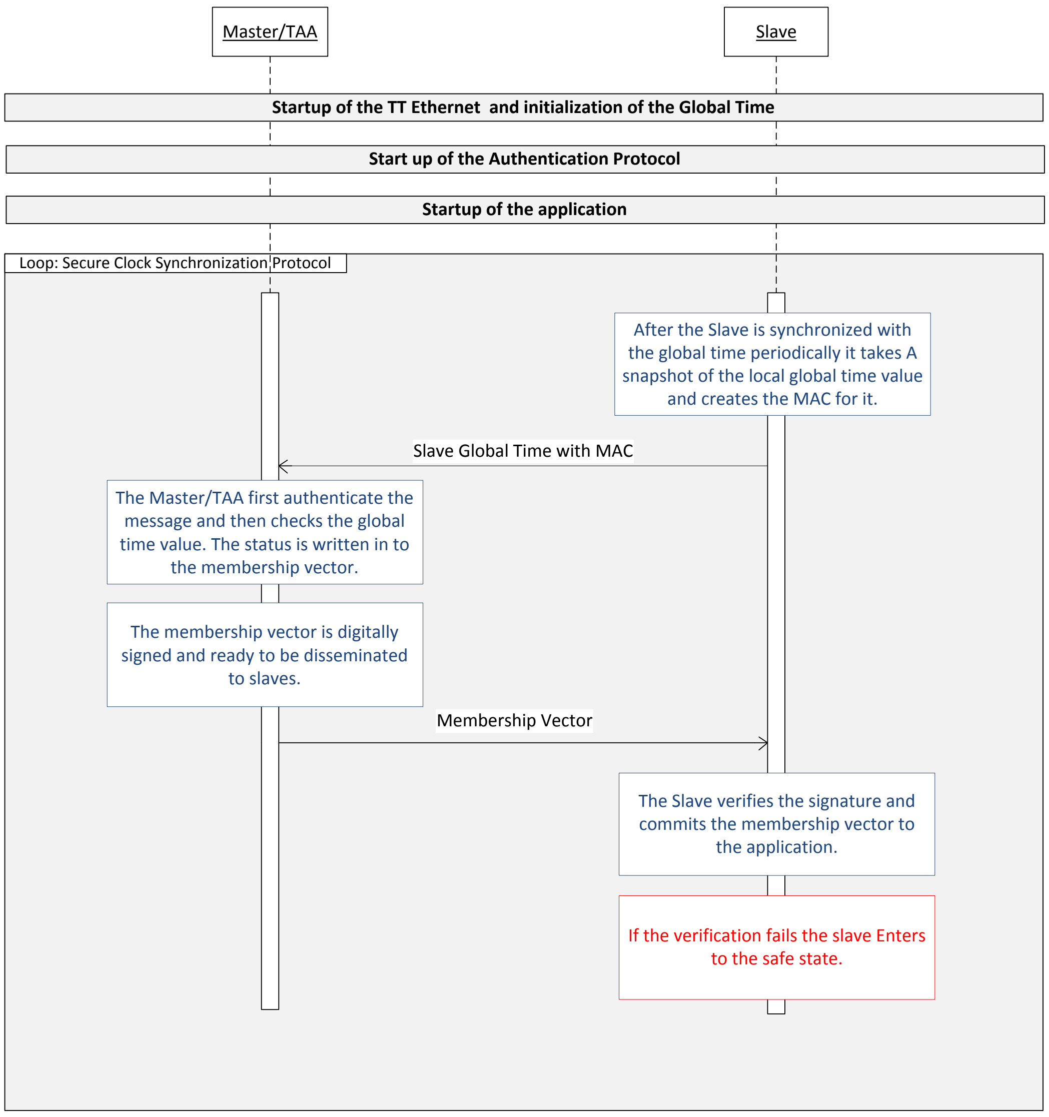

Master/TAA
Slave
Startup of the TT Ethernet and initialization of the Global Time
Start up of the Authentication Protocol
Startup of the application
Loop: Secure Clock Synchronization Protocol
After the Slave is synchronized with the global time periodically it takes A snapshot of the local global time value and creates the MAC for it.
Slave Global Time with MAC
The Master/TAA first authenticate the message and then checks the global time value. The status is written in to the membership vector.
The membership vector is digitally signed and ready to be disseminated to slaves.
Membership Vector
The Slave verifies the signature and commits the membership vector to the application.
If the verification fails the slave Enters to the safe state.

Figure 4.4: Sequence diagram of the secure clock synchronization protocol.

slave nodes in a digitally signed message. Hereby the second stage of the protocol in which detection of the malicious faults takes place is over. In the third stage the detected malicious faults are committed to applications on slave nodes. First, the slaves verify the signature of the message with the public key of the master. If the verification of the signature fails, the slave recognizes that the master is fraud and that the synchronization can not be trusted. If the synchronization can not be trusted the basic condition for the time triggered communication is not fulfilled and the slave goes to a safe state. If the verification of the digital signature is successful, the membership vector contained in the message is copied to the designated place and committed to the application. The application routines check the status of the current node and if the slave is marked as untrusted in the membership vector, the slave is brought in to a safe state. The application also has access to the statuses of other nodes and if it notices that some of the nodes with whom it communicates can not be trusted, it can terminate the communication and wait until a valid synchronization is established.

The visual representation of the protocol is depicted on the sequence diagram in Figure 4.4. The cryptographic tools MAC and DSA used in the Figure 4.4 are chosen for simpler understanding of the protocol functionality. Also other cryptographic tools with the same security properties could be used instead of these cryptographic methods.

4.3 Summary of the System Model

The advantage of using the secure synchronization protocol in TTE is a secure and dependable synchronization within TTE network. The increase of the security level increases also the safety of the system. This augments the area of the utilization for the TTE. The downside of the protocol is the requirement of additional resources. Embedded systems are usually designed to maximise the usability of the available resources. On the other hand, it is definitely an acceptable trade-off for systems where security means safety.

CHAPTER 5

Implementation

In Chapter 4 the concept of the secure synchronization protocol has been described. This chapter describes the practical implementation of the system for development and evaluation of the secure clock synchronization protocol on TTE . The appropriate system must be built in order to test the model and its characteristics . The system must be able to simulate malicious faults which can endanger the global time and execute the authentication algorithm to counteract them. To realize these tasks a chain of tools is required starting from the hardware components, development software, application software to the authentication protocol itself. The implementation of the secure clock synchronization protocol for Time Triggered Ethernet comprises several development tools and methods:

- hardware,
- system software,
- development, debugging and evaluation tools.
- application software,

In the following sections all tools and methods will be described in detail.

5.1 Hardware

In Section 3.1 a basic real-time system architecture has been described. Later on in the Section 3.3 this same architectural concept has been described in the scope of TTE. According to this concept, two essential elements in this architecture are the host computer and the communication controller. In most cases the former integrates the latter forming a single computational unit. For the purposes of this thesis a small embedded development computer named Soekris net4801 [32] has been used as the host computer. The communication controller is a TTE Controller implemented as an PCI extension card.

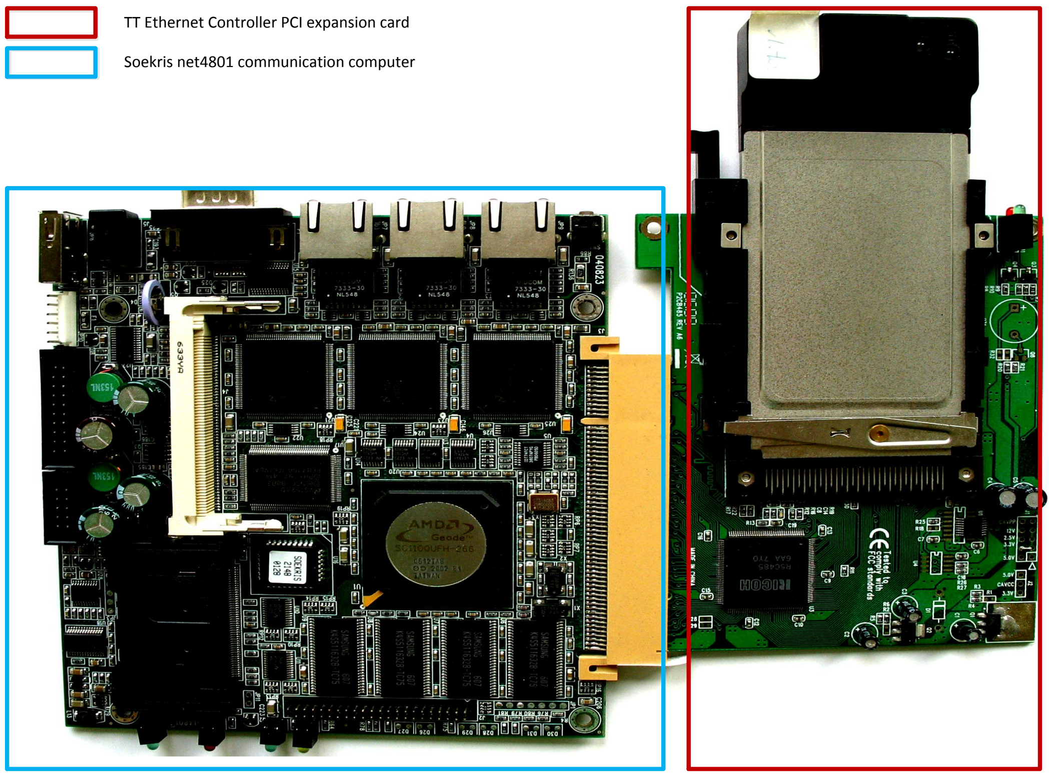

TT Ethernet Controller PCI expansion card
Soekris net4801 communication computer

Figure 5.1: Development board Soekris net4801 with TTE Controller PCI extension card.

5.1.1 Soekris net4801

The communication computer Soekris net4801 [32] is based on the GEODE SC1100 266 MHz [33] embedded processor from Advanced Micro Devices (AMD). The system is PC [1] compatible designed for network and communication applications [32]. The net4801 communication computer disposes with the 32 - 128 MB PC133 SDRAM [2] and 512 KB Flash memory for BIOS[3] purposes. The communication interface consists of two PCI [4] slots, standard PCI and Mini-PCI [5] ports, three standard Ethernet ports, a USB [6] port, two serial ports. The board also disposes with twelve general purpose I/O pins for various uses. For the purposes of the mass storage device there is a one CompactFlash type I/II socket. The installation of a operation system on Soekris

[1]Personal Computer
[2]Standard for Synchronous dynamic random memory
[3]Basic input/output system
[4]Peripheral Component Interconnect Bus
[5]Miniature version of PCI designed for uses on systems with dimension constraints.
[6]Universal Serial Bus

net4801 computer can be done using CompactFlash or using a standard Ethernet connection. More on this subject will be discussed in Section 5.2.

5.1.2 TTE Controller

The communication controller used in this implementation is designed at the Technical University of Vienna by the Real Time Systems Group [34].The TTE Controller is implemented as a Cardbus [7] expansion card. The configuration and data area are mapped into the memory of the host computer, in this case Soekris net4801. The functionality of the card is implemented as FPGA [8] with Altera Cyclone II [35] device. The standard Ethernet transceiver is used to convert the I/O standard of the FPGA to the Ethernet physical level standard. Each card can be uniquely identified with the configuration data stored in a non-volatile memory of the card.

5.1.3 Server and Network Equipment

To complete the description of the hardware part of the development environment the network equipment and development server is yet to be described. The network equipment used to implement the secure clock synchronization protocol on TTE comprises three communication networks: the standard Ethernet detached from TT Ethernet, the RS-232 network and the TT Ethernet network. The first two networks connect TTE nodes with a server and are used for the development, diagnostic purposes and the evaluation. The server is a standard computer (PC) specially equipped with hardware components and software tools for development, communication, diagnosis and evaluation.The TT Ethernet network is used for the implementation and the testing of the Secure Clock Synchronization Protocol. Except of the server networks communicate over switches for standard Ethernet, RS-232 and TTE. The TTE Switch is specially designed to be able to handle both the time-triggered and standard Ethernet communication and it is also implemented in FPGA.

5.2 System Software

All essential hardware components of the development environment have been described in Section 5.1. System software required for operation of hardware components consists of operating systems and drivers for hardware components. Two types of operating systems are required for the implementation of the secure clock synchronization protocol:

- real-time operating system for the Soekris net4801 computers, and
- standard Linux equipped with the additional server infrastructure and the appropriate driver software for the TTE Controller.

The driver software enables operating system to control the TTE Controller by creating memory mapping between the host computer and the TTE Controller.

[7]Standard for PCMCIA 5.0 or latter devices, which is 32-bit PCI bus in form of a card extension. `http://web.archive.org/web/20080822091330/http://www.pcmcia.org/`

[8]Field Programmable Gate Array

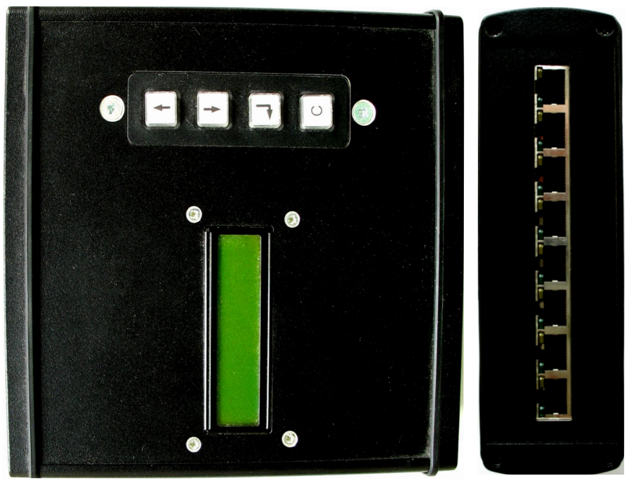


Figure 5.2: TT Ethernet Switch implemented in FPGA.

5.2.1 Operating System for Soekris net4801

The TTE Node is designed as a real-time fault-tolerant system with the ability of real-time communication.The design ensures that the system provides temporal predictability and determinism. To achieve these goals, both the TTE Controller and the host computer must be implemented with these requirements. On the one hand, the controller for TTE is the dedicated component designed to provide real-time fault-tolerant communication. On the other hand the host computer, in this case Soekris net4801, is general purpose communication computer and requires specially designed software to provide temporal predictability and determinism. This is why the use of a real-time operating system is necessary in these systems.

The real-time operating system used within this thesis is based on Linux with changes in kernel to provide hard real-time task execution. The Linux distribution is patched with the Real Time Application Interface (RTAI) creating an environment where standard basic functions and hard real-time functions are available. The patching of the Linux kernel is done by installing a generic Real Time Hardware Abstraction Layer (RTHAL). The RTHAL reorganizes the kernel internal data and functions dependent so that they can be used by RTAI when hard real-time is needed [36]. The RTAI provides task schedulers and services needed to provide temporally predictable and deterministic execution. The Linux distribution used to create real-time operating system is called Voyage [37] . It is a Debian [9] based Linux distribution specially optimized for embedded systems. It requires very little resources and still supports almost all standard Linux functionalities. This makes it a ideal operating system for a communication node. For the implementation of TTE Node within this thesis Voyage Linux is patched with the RTAI to create a light fully functional Linux based operating system. It provides full hard real-time POSIX

[9] Linux distribution for x86 and embedded architectures. `http://www.debian.org/`.

[10,11] support for kernel threads and RTAI tasks, which enables efficient partitioning of tasks into threads. This way it is easier to separate application and authentication protocol within a program.

Installation of the operating system on Soekris net4801 boards can be performed either by loading the system image on the Compactflash card and booting the system from the card directly, or storing it on the development server and booting the system over the network. The second option is more convenient for the application development because the whole file system of the target system (host computer of the TTE Node Soekris net4801) is placed on development server and can be accessed without any restrictions. It is therefore obvious choice for development of a distributed system where developer must simultaneously install application software on several target systems.

The last step of system software setup is to install driver software for the TTE Controller. Without the driver software a TTE Application can not be executed. The driver software is implemented as a standard Linux kernel module and as such it is executed using standard Linux procedures. It maps register files of the TTE Controller to the memory of the host computer. The driver software is written in the generic C code and can be adopted for various platforms.

Boot process for Soekris net4801 and Voayage/RTAI Linux

The boot-loader of the Soekris net4801 is implemented with included drivers for the standard Ethernet and it can establish connection with the server, which provides the kernel image of the operating system and boot the system over the network.

The boot-loader uses Preboot Execution Environment (PXE) to load the operating system into the on-board memory. PXE is a procedure designed to allow over-the-network-booting of the client system in an enterprise.This allows a better control over large number of computer systems and simplifies development and maintenance. PXE uses standardized network protocols and communication methods. The PXE system consists of a slave node or a target node and a server on which file systems and images of the slaves are stored. The Server also hosts services needed for the execution of network protocols. In the first step of the PXE protocol the target node searches for a valid DHCP [12] server, if the server is reachable, it assigns IP address to the target system and the boot-loader can acquire information about a running PXE [38] server in the network. The PXE server provides the target system with the file path to the Network Bootstrap Program (NPB), in this case the image of the Voyage/RTAI Linux kernel, and loads it to the RAM memory of the target computer. The NPB is transferred using a Trivial File Transfer Protocol (TFTP) [39]. The kernel is compiled with the ability to use a Network File System (NFS) [40] instead of the traditional disk based file system. This avoids the need for a mass storage devices on the system and simplifies the usage.

[10]Portable Operating System Interface (POSIX) between operating system and application.

[11]`http://pubs.opengroup.org/onlinepubs/9699919799/`

[12]Dynamic Host Configuration Protocol

5.2.2 Operating System for Development Server

For a development server a Debian Linux distribution is chosen as the operating system. The installed version of the Linux is adapted to serve as the server for several communication protocols needed by the target systems such as booting and services used in development toolchain. It provides networking services which allow a user great flexibility in development, debugging and evaluation. The networking services installed on the system are: DHCP server service, FTP server service, TFTP server service, SSH server service, RS-232 communication software. The application of these services for the booting process of the target systems are described in Section 5.2.1, a use of the services for other purposes is described in latter sections.

5.3 Development, Debugging and Evaluation Tools

For the implementation of the secure synchronization protocol two types of the software are used: a system software described in Section 5.2 and a application software. The application software comprises all programs created by the user to accomplish a given task. In the case of this thesis the task is to establish the time triggered communication using TTE with a secure clock synchronization algorithm. The development process for the application software is implemented in four stages:

- source code writing,
- compiling and linking,
- installation and execution,
- debugging and evaluation.

The application software is written with a C programming language and the standard GNU Compiler Collection (GCC) is used for the compiling and linking. The applications are executed on the Soekris net4801 computer with the Voyage/RTAI Linux operating system platform. These computers are limited with computational power and development process is carried out on development server. The compilation of the source code is done using a cross compiler so that environment variables and software library dependencies correspond to the Voyage/RTAI Linux platform. To simplify the development of the application software, the whole process is integrated into specially adapted software named Eclipse IDE [13] [41].

The first three steps of the development process are executed on the development server as the final development stage is implemented on the target computers. All three initial steps can be executed within the Eclipse IDE. The source code is translated by the compiler to an executable file which is than installed on the target system. The installation requires transfer of the binary executable files to a specially designated places on the hard drive of the server, where the file systems of the target operating systems are stored. The content of the target file system can be changed by the server at any point in time even while the target computer is running. This makes development of the application software for the target system very simple and effective.The execution of a application is preformed by a user on the target using the standard Linux console.

[13]Integrated Development Environment

The console of the target system can be accessed either using a RS-232 connection with the Minicom [14] program on the server side or using a SSH connection. The SS connection can be made directly from the server or trough the server which is connected to the external network from the remote workstation.

The lasts steps in the development process are to find potential errors and to evaluate results that were achieved with the secure clock synchronization algorithm. The debugging of the application is done using three methods:

- monitoring kernel diagnosis console output using minicom or ssh,
- monitoring stdio error messages,
- monitoring TTE communication channels using Wireshark[15].

The evaluation of the secure clock synchronization protocol requires precise temporal measurements. To provide both visual and statistical evaluation of the protocol the system is connected with a digital oscilloscope over a general purpose I/O (GPIO) port on the Soekris net4801 computer. To enable GPIO control additional drivers must be installed for the hardware controllers which control the GPIO on the Soekris net4801. A corresponding device for each pin must be identified and installed in the Voyage/RTAI Linux. The controlling I/O pins to show the temporal behaviour of the TTE messages out of the application software create delays which taint the results. Instead of modifying the application software, the drivers for TTE are modified to avoid delays, so that they produce signal on I/O pins for certain event. This event can be: a reception interrupt, a transmission interrupt, a periodic host interrupt etc.. This method for the evaluation of the temporal behaviour of the TTE system captures events with μs precision. The signals produced over GPIO are captured with an oscilloscope for further analysis and statistical evaluation. More on this topic and the results of the evaluation are presented in Chapter 6.

5.4 Application Software

The standard TTE application designed for the RTAI Linux operates trough RTAI tasks. In this implementation each task occupies a single POSIX thread. The main() function of such application is used for a initialization and a configuration while the standard functionality is organized as a state machine in the single RTAI task. This task is executed periodically and it communicates with the driver of the TTE controller trough a mailbox system characteristic for RTAI Linux. The data received from the driver are reports on the ongoing events in the hardware (i.e. message reception interrupt). The state machine uses these messages for the hardware and controls accordingly the program .The application specific functions can be implemented within the thread as a function or as an additional thread. The implementation of the secure clock synchronization protocol requires parallel execution of its routines. Therefore, it is implemented in two additional threads. In the Figure 5.3 the structure of the application software has been depicted including the authentication protocol threads.

[14]Minicom is a text based terminal emulation software for modem control. Mostly used for establishing a serial console connections. `http://alioth.debian.org/projects/minicom/`

[15]Wireshark packet analyser program used for network development. `http://www.wireshark.org/`

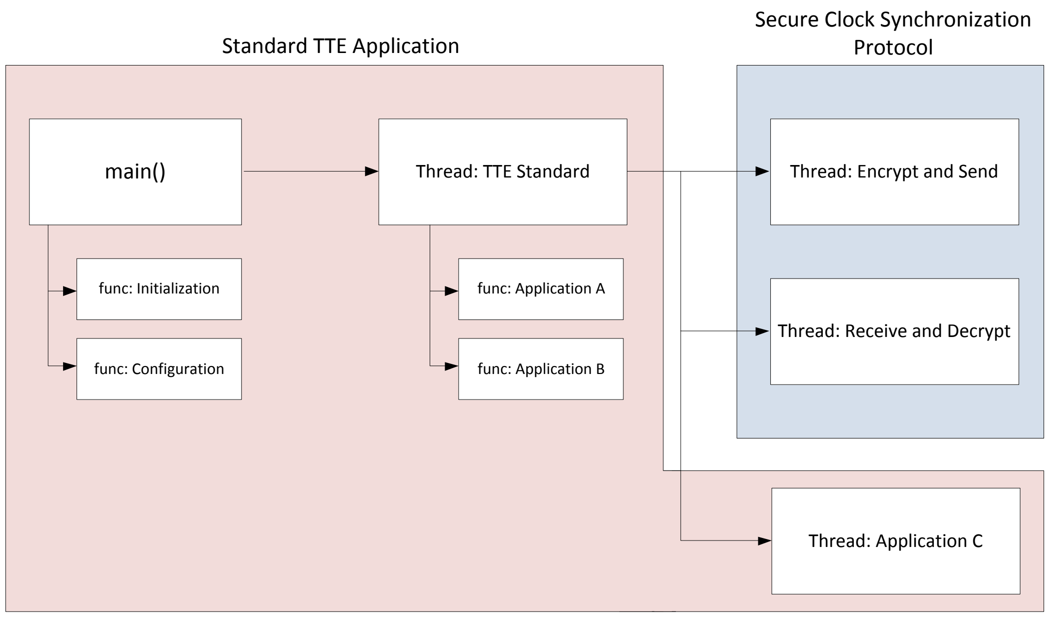


Figure 5.3: The software structure of the implementation of the secure clock synchronization protocol. The red block depicts the TTE application without the secure clock synchronization protocol. The blue block contains tasks required for the execution of the authentication protocol.

5.4.1 The Secure Clock Synchronization Protocol

The integration of the authentication protocol algorithm in to the TTE is done by adding its routines in to the application software of the TTE system. The structure of the modified application is described in the introduction of the Section 5.4. The authentication protocol is implemented in two POSIX threads as a hard-real time RTAI task. The first thread is generally called **encrypt and send**. On the master side it is named **sign and update** and its task is to apply the cryptographic schemes the membership vector and to update the data with the cryptographic context in to the CNI memory. On the master side this thread is called **authenticate and update** and its task is to authenticate local time of a slave and to update the CNI memory. The second thread is named **receive and verify** thread and its task is to verify received messages and to commit the information to the application. On the master side it is called **Verify Slave Time** and on the slave side **Verify Membership Vector**. The schedule of the TTE application must also be modified to accommodate messages needed by the authentication protocol. The security properties of the authentication protocol are implemented via cryptographic tools incorporated into a cryptographic library named libtomcrypt [29] [42] [43]. The One-key Message Authentication Code (OMAC) [44] algorithm is used for the message authentication, and the Eliptic Curve DSA (ECDSA) [30] for the digital signature of the messages.

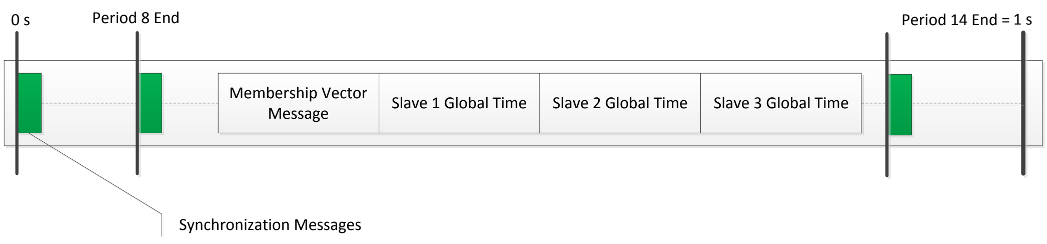


Figure 5.4: The relative placement of the messages required by the secure clock synchronization protocol in the application schedule.

The Schedule

The schedule of the TTE system determines the temporal execution of the tasks. It is defined by the application and it requires a slot for the synchronization messages other messages are optional.Beside messages, the schedule can also contain user defined periodical interrupts called Host Interrupts. These interrupts can be used to schedule a periodical task for the application. Although each period can be chosen for the synchronization message, shorter periods are more convenient because of the better precision. A application with the authentication protocol requires two more types of messages. The one is for the membership vector from the master to the slaves and the other is for the global time messages from the slaves to the master. The authentication protocol messages are scheduled in the longer period, like the period 14 in order to leave enough time after each transmission for the processing of data and a execution of the cryptographic tasks. The Host Interrupt is used to execute the needed tasks for the protocol like the authentication or digital signing. Chronologically, the Host Interrupt is set at the beginning of the period 14 with the phase offset to avoid the conflict with the synchronization message and to leave enough time for the cryptographic tools to be executed on messages before the message sending slot. The depiction of the schedule in Figure 5.4 shows a relative placement of the messages within the interval of one second.

The facts behind the schedule are summarized in the Table 5.1.

Message	Period (dec.) [15]	Phase (hex.)[16]
Synchronization Message	1	$0x000$
Host Interrupt	14	$0x006$
Membership Vector Message	14	$0x802$
Global Time Message Slave 1	14	$0x803$
Global Time Message Slave 2	14	$0x804$
Global Time Message Slave 3	14	$0x805$

Table 5.1: Authentication Protocol Schedule

The Algorithm

In this section the algorithms used to implement the authentication protocol are described. The algorithms presented below are written in pseudo code.There are two algorithms: a master algorithm and a slave algorithm. The algorithms are simplified in comparison to the actual C code for better understanding. Nevertheless they posses all essential characteristics, needed to show the functionality of the secure clock synchronization protocol.

The algorithms contain several important variables which control the execution of the protocol. First, the boolean variable $terminate$ controls the execution of the entire program. As long as this variable is set to $false$,the program continues to run periodically, if the variable is set to $true$,the program will terminate freeing all resources. Each interrupt has a unique identification number. The variable $ID_{interrupt}$ stores the interrupt identification number in case of the host interrupt or error interrupt, or the message identification number in case of a message reception interrupt. The $ID_{ecryptinterrupt}$ contains the identification number of the host interrupt defined for the authentication protocol. $(ID_{message})$ holds the unique identification number of the message which is received . $Auth_{ID}$ is the set of all message identification numbers of the messages that are send by the slaves and are related to the authentication protocol. The identification number of the message that is sent from the master to slaves and that contains the membership vector, is represented with the variable $ID_{MembershipVectorMessage}$. Each node in the TTE network has also unique identification number (ID_{Slave}). The master uses this id to select the corresponding symmetric key for a given slave. The membership vector is represented with the variable $MembershipVector$. The boolean variables $untrusted$ and $trusted$ represent the status of the slaves within membership vector. The time is represented with the $GlobalTime_{Local}$ for the local global time of the master and $SlaveSendInstant$ for the local clock value of the slave at the send instant of the message from a slave to the master. $Signature$ is the digital signature of the membership vector calculated by the master via ECDSA, and MAC is the message authentication code calculated by the slaves via OMAC.

Algorithm 1 represents the master node implementation of the authentication protocol algorithm. The purpose of the master node in the authentication protocol is to serve as a trusted authentication authority. It is responsible for gathering global time values, verifying them and informing the slaves about the security status of the cluster. On the master side of the protocol there are two tasks:

- The **sign and update** task is described in Algorithm 2 and is responsible for signing the membership vector messages and for updating the CNI memory with the new membership vector message. The digital signature is created with ECDSA using a secret private key of the master. The TTE controller automatically transmits the membership vector message in CNI memory to the slaves in the next round. The signing of the membership vector and the updating of the membership vector message is done periodically on the occurrence of encrypt interrupt which generated by the TTE controller.

[15] Decimal number representation.

[16] Hexadecimal number representation.

Algorithm 1 The secure clock synchronization algorithm of the master node.

```
1: procedure MAIN CONTROL( )
2:     $PrivateKey \leftarrow$ GET PRIVATE KEY FOR ECDSA( )
3:     $Auth_{ID} \leftarrow$ GET MESSAGE IDs FOR AUTHENTICATION PROTOCOL( )
4:     while $terminate$ = false do
5:         $ID_{interrupt} \leftarrow$ WAIT FOR INTERRUPT( )
6:         if $ID_{interrupt} \in Auth_{ID}$ then
7:             TRIGGER RECEIVE AND VERIFY TASK($ID_{interrupt}$)
8:         else if $ID_{interrupt} = ID_{ecryptinterrupt}$ then
9:             TRIGGER SIGN AND UPDATE TASK( )
10:        else
11:            STANDARD INTERRUPT HANDLER($ID_{interrupt}$)
12:        end if
13:    end while
14: end procedure
```

Algorithm 2 Sign and Update Task of the master node

```
1: procedure SIGN AND UPDATE( )
2:     $Signature \leftarrow$ SIGN ECDSA($MembershipVector$, $PrivateKey$)
3:     $Message \leftarrow$ MERGE($Signature$,$MembershipVector$)
4:     UPDATE MESSAGE($Message$)
5: end procedure
```

Algorithm 3 Verify Slave Time Task of the master node

```
1: procedure VERIFY SLAVE TIME($ID_{interrupt}$)
2:     $ID_{message} \leftarrow ID_{interrupt}$
3:     $ID_{Slave} \leftarrow$ GET SLAVE ID($ID_{message}$)
4:     $SymetricKey \leftarrow$ GET SYMMETRIC KEY($ID_{Slave}$).
5:     $Message \leftarrow$ GET MESSAGE($ID_{message}$)
6:     $SlaveTime_{SendInstant} \leftarrow$ GET SLAVE TIME($Message$)
7:     $MAC_{message} \leftarrow$ GET MAC($Message$)
8:     $MAC \leftarrow$ CREATE MAC($SlaveSendInstant$, $SymetricKey$)
9:     $GlobalTime_{Local} \leftarrow$ GET GLOBAL TIME LOCAL( )
10:    if $MAC = MAC_{message}$ & VERIFY($GlobalTime_{Local}$,$SlaveSendInstant$) then
11:        $MembershipVector \leftarrow$ SET STATUS($ID_{Slave}$,$trusted$)
12:    else
13:        $MembershipVector \leftarrow$ SET STATUS($ID_{Slave}$,$untrusted$)
14:    end if
15: end procedure
```

- The second task described in Algorithm 3 is called **Verify Slave Time** and it is responsible for the reception of the global time messages from the slaves and the authentication and the verification of the global time. The global time messages are authenticated by the slaves with OMAC and each master/slave pair share a single secret key for authentication. The verification of the global time is performed by comparing the global time value received from a slave with the local global time of the master. If the results of the authentication or verification are negative the slave is marked as untrusted in the membership vector.

Algorithm 4 The secure clock synchronization algorithm of a slave node.

```
1: procedure MAIN CONTROL( )
2:     $PublicKey \leftarrow$ GET PUBLIC KEY FOR ECDSA( )
3:     $SymetricKey \leftarrow$ GET SYMMETRIC KEY($LocalSlaveID$).
4:     while $terminate$ = false do
5:         $ID_{interrupt} \leftarrow$ WAIT FOR INTERRUPT( )
6:         if $ID_{interrupt} = ID_{MembershipVectorMessage}$ then
7:             TRIGGER RECEIVE AND VERIFY TASK($ID_{interrupt}$)
8:         else if $ID_{interrupt} = ID_{encryptinterrupt}$ then
9:             TRIGGER AUTHENTICATE AND UPDATE TASK( )
10:        else
11:            STANDARD INTERRUPT HANDLER($ID_{interrupt}$)
12:        end if
13:    end while
14: end procedure
```

Algorithm 5 Authenticate and Update Task of a slave node

```
1: procedure AUTHENTICATE AND UPDATE( )
2:     $GlobalTime_{Local} \leftarrow$ GET GLOBAL TIME LOCAL( )
3:     $MAC \leftarrow$ CREATE MAC($GlobalTime_{Local}$)
4:     $Message \leftarrow$ MERGE($MAC$,$GlobalTime_{Local}$)
5:     UPDATE MESSAGE($Message$)
6: end procedure
```

Algorithm 4 represents the slave node implementation of the authentication protocol algorithm. It is responsible for delivering its local view of global time in an authenticated manner to the master. Like the master, the slave algorithm can be divided in two basic tasks:

- The task described in Algorithm 5, the **authenticate and update** task, provides authentication of the global time messages which are sent to the master. The authentication is implemented using the OMAC algorithm, by creating a MAC from the global time value and a secret key.

- The second task is named **Verify Membership Vector** (Algorithm 6). It verifies the membership vector messages received from the master and provides the received membership

Algorithm 6 Verify Membership Vector Task for a slave node

```
1: procedure VERIFY MEMBERSHIP VECTOR($ID_{interrupt}$)
2:     $Message \leftarrow$ GET MESSAGE($ID_{MembershipVectorMessage}$)
3:     $MembershipVector_{message} \leftarrow$ GET MEMBERSHIP VECTOR($Message$)
4:     $Signature_{ID_{MembershipVectorMessage}} \leftarrow$ GET SIGNATURE($Message$)
5:     $Status_{Sigrnature} \leftarrow$ VERIFY SIGNATURE($Signature_{ID_{MembershipVectorMessage}}$,$PublicKey$)
6:     if $Status_{Sigrnature} \neq valid$ then
7:         $terminate \leftarrow true$                ▷ The message is compromised by an attacker.
8:     else
9:         $MembershipVector_{local} \leftarrow MembershipVector_{message}$
10:                        ▷ The membership vector is committed to the application.
11:    end if
12: end procedure
```

vector to the application. The membership vector message contains a digital signature generated with the ECDSA. To verify the digital signature a slave uses the corresponding public key. Only if the signature is verified successfully, the membership vector is made available to the application.

The functions used in the algorithms are explained in the following text:

- **Trigger Sign and Update Task ( )** starts the task for signing the membership vector message updating the corresponding location in the CNI. This functionality was realized as a separate task to keep the interrupt service routine as short as possible. Thus, the master can receive other messages while this executes.

- **Trigger Authenticate and Update Task ( )** starts the task for authenticating the message containing the slave's local view on the global time for updating the corresponding location in the CNI. This functionality was realized as a separate task to keep the interrupt service routine as short as possible. Thus, a slave can receive other messages while this executes.

- **Trigger Verify Slave Time Task (*Interrupt ID*)** starts the task for the verification of the received global time message from the slave. This functionality was realized as a separate task to keep the interrupt service routine as short as possible. Thus, the master can receive other messages while this executes.

- **Trigger Verify Membership Vector Task (*Interrupt ID*)** starts the task for the verification of the received global membership vector message from the master. This functionality was realized as a separate task to keep the interrupt service routine as short as possible. Thus, a slave can receive other messages while this executes.

- **Get Public Key for ECDSA( ), Get Private Key for ECDSA( )** provide keys for the ECDSA.

- **Get message IDs for Authentication Protocol( )** creates set of the message identification numbers from the messages related to the authentication protocol.
- **Update Message(*Message*)** copy the message content to the designated place in the memory which is than sent over the network.
- **Create MAC(*Global Time*)** is the function for the message authentication. Same function is used for the encryption on the slave side and the verification on the master side.
- **Merge (*Data 1, Data 2*)** merges two data blocks in to the single message.
- **Get Message(*Message ID*)** retrieves message data for the interrupt ID received from the controller over RTAI mailbox.
- **Get MAC(*message*** extracts MAC from the message.
- **Standard Interrupt Handler (*ID*)** represents handler function for the interrupts in the standard application.
- **Get Slave Time (*message*)** extracts the local clock value of the slave from the received message.
- **Get symmetric key (*Slave ID*)** retrieves the symmetric key for the slave using its identification number.
- **Get Global Time Local**( ) returns the local value of the global time.
- **Set Status (*slave, status, membership vector*)** writes current security status for the given slave in to the membership vector.
- **Verify(Time 1, Time 2)** compares two time values and returns *true* if the drift offset between those two values is within permitted boundaries and *false* otherwise.
- **Sign ECDSA (*payload, private key*)** creates digital signature from the plaintext using private key.
- **Get Signature(*Message ID*)** retrieves the signature from the membership vector message.
- **Verify Signature(*signature, public key*)** provides verification of the signature for the ECDSA. The function returns status of the signature (e.g. valid, invalid).
- **Get Membership Vector ( )** retrieves the membership vector received from the master.
- **Wait for Interrupt( )** blocks the execution by waiting on a interrupt from the TTE Controller. The function returns the interrupt ID in case of the interrupt and message ID in case of the message reception interrupt.

One-key Message Authentication Code (OMAC)

The OMAC is the version of block-cipher chaining message authentication code (CBC MAC) [45]. It is designed to accept the messages of any length and its performance is much more optimized than the standard CBC MAC.It is also recommended by the NIST in their publication on cipher modes of operation [46]. The recommended cipher in this publication is named CMAC and it corresponds to the OMAC1 implementation. The same version is implemented in a libtomcrypt collection of cryptographic tools.This implementation uses the AES symmetrical cipher with a 128-bit key to create MAC. This combination is considered to be safe for a foreseeable future. The keys are stored statically without any key management system which assumed that keys are secure. The execution time of the OMAC algorithm form the libtomcrypt is under 0.01 seconds on the Soekris net4801 with the RTAI Voyage Linux.

Elliptic Curve Digital Signature Algorithm (EC DSA)

The EC DSA is a variant of the DSA using an elliptic curve public key cryptography. In comparison with the standard DSA key, lengths are much shorter for the same security level, which brings significant performance increase. This property is valuable when the algorithm is implemented on embedded systems with the limited computational power. For the implementation of this thesis 224-bit long keys have been used, the recommended length by the NIST is 256-bit [30]. For the embedded applications can be used even shorter keys, as the life span of the data is relatively short (i.e. sensor data).

5.5 Summary of the Development Environment

In this chapter all components of the development environment have been described and their role in the overall system has been established. The platform created to support the implementation phase of this thesis has a multifunctional character and it can be easily adapted for other research projects. The schematic overview on the development environment is depicted in Figure 5.5. The combination of the security model presented in this thesis with other security methods creates a cornerstone for a security platform for the time triggered architecture.

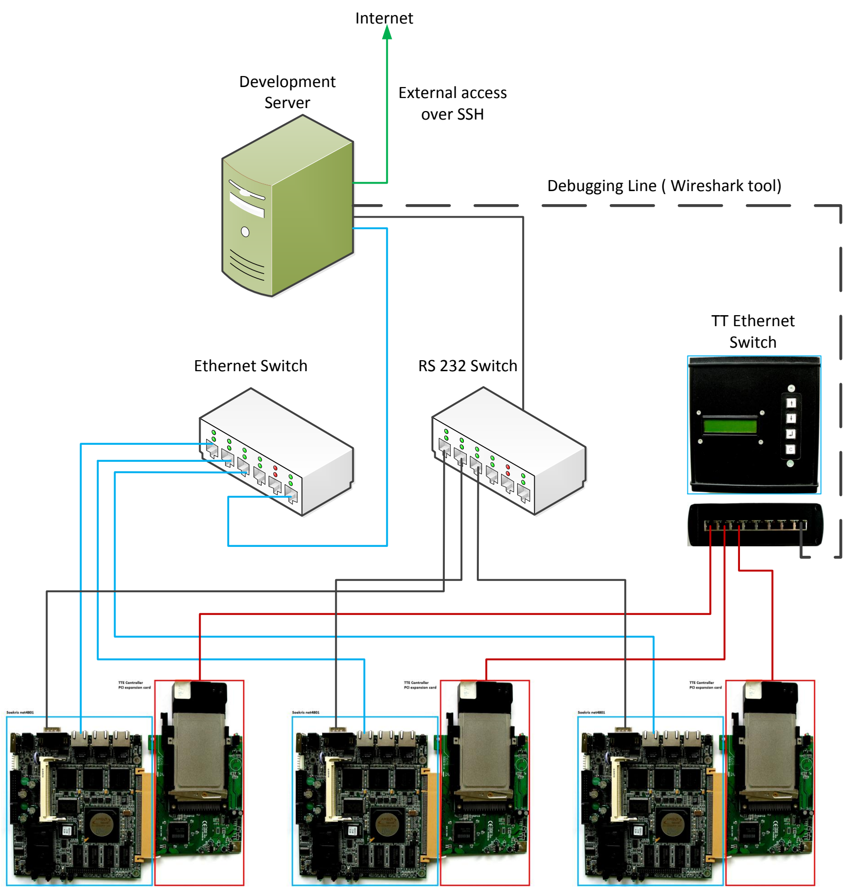


Figure 5.5: The schematic depiction of development environment. The blue lines represent standard Ethernet circuit used for development, booting of the target system and debugging. Black lines represent RS-232 network circuit used for the control and debugging. Red lines are the TT Ethernet circuit. The dashed black line and the green line are the debugging connection for the TT traffic and the connection to Internet.

CHAPTER **6**

Experimental Evaluation

For the evaluation of the secure clock synchronization protocol we will simulate an attack on the timing of the clock synchronization messages that are received by the attacked host. In the next sections we describe the attack scenario, the experimental setup, and the experiments themselves.

6.1 Attack Scenario

According to the attacker model described in Section 4.1 we assume that an attacker has the ability to change the timing of all messages on the network in two ways, it can delay messages or it can speed up messages. In the case of clock synchronization messages, this will lead in both cases to a synchronization error of the attacked node (i.e., the node that receives the delayed or speeded up messages), if the timing modification is not detected.

To a given extend, TTE is natively robust against such attacks. If the timing of a clock synchronization message is modified such that it is received outside the reception window of the node, the node will detect the modification and will not accept the synchronization message. In our scenario, the attacker performs a very clever attack, that makes it impossible for the TTE controller alone to detect the timing modification. This is possible, by modifying the timing of all messages received by a node only by a such small amount that they stay in the bounds of the node's reception window. In this way, the node will not be able to detect the timing modification. Furthermore, it will adjust its clock value to the modified timing of the received clock synchronization messages (keep in mind that the attacker delays or speeds up all messages that are received by the attacked node). Since the attacked node has adjusted its local clock value, the reception window will have slightly moved towards the direction of the timing modification in the next resynchronization interval. Thus the attacker can now increase the magnitude of the timing modification of the received messages (whether it was delayed or speeded up) without being detected. With each received clock synchronization message, it can slightly increase the delay (or speed up) without violating the reception window and thus cause the attacked node to drift away from the cluster without being detected by the attacked node, i.e. the attacked node cannot recognize that it is no more in synchrony with the rest of the cluster.

6.2 Experimental Setup

The development environment is connected to an oscilloscope for the experimental evaluation of the implemented concepts. The clamps of the oscilloscope are connected to the GPIO pin on the Soekris net4801 computer. This pin is set or cleared by the TTE controller in order to signalise a system-wide periodic instant, with respect to the local clock of a node. If the cluster would be always perfectly synchronized, this signal would periodically change its value at all nodes at exactly the same time. The maximum offset of this signal for any two nodes indicates the actual precision of the cluster. The measurement data are stored on the oscilloscope locally, or on a network drive. We use Matlab for statical evaluation and for the representation of the results.

The TTE cluster used for experiments consists of a master node and three slave nodes. The master node provides the global time to the cluster and serves as the trusted authentication authority. The period for the synchronization message is 8 with the phase 0. The host interrupt is scheduled in period 14 with the phase $0x006$.

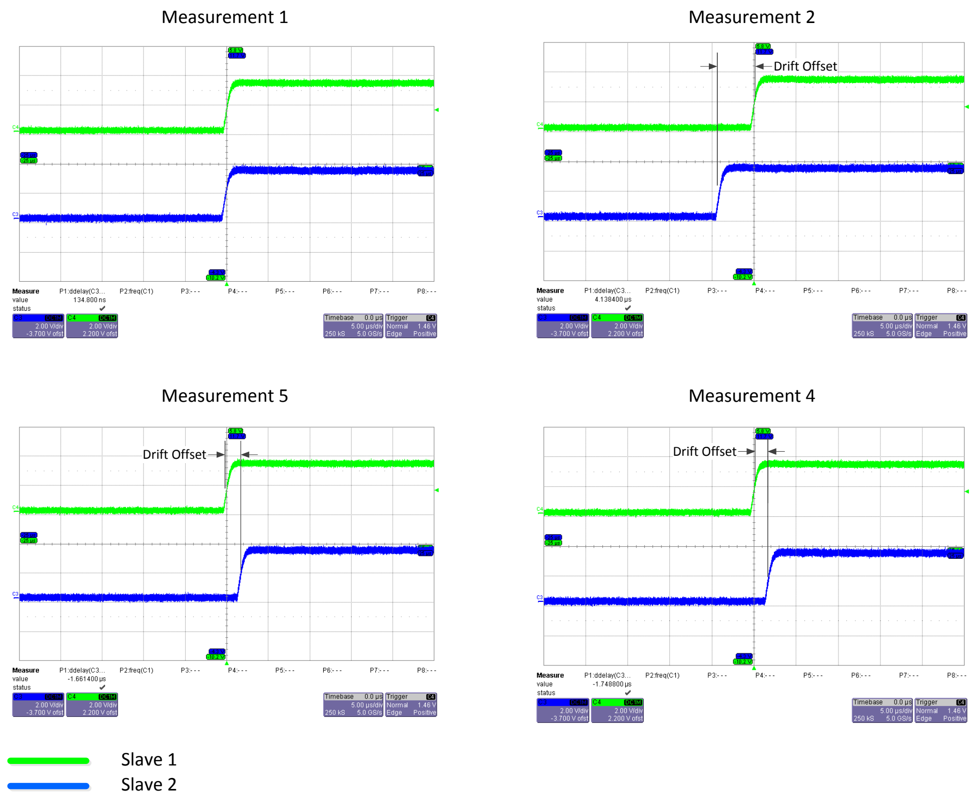


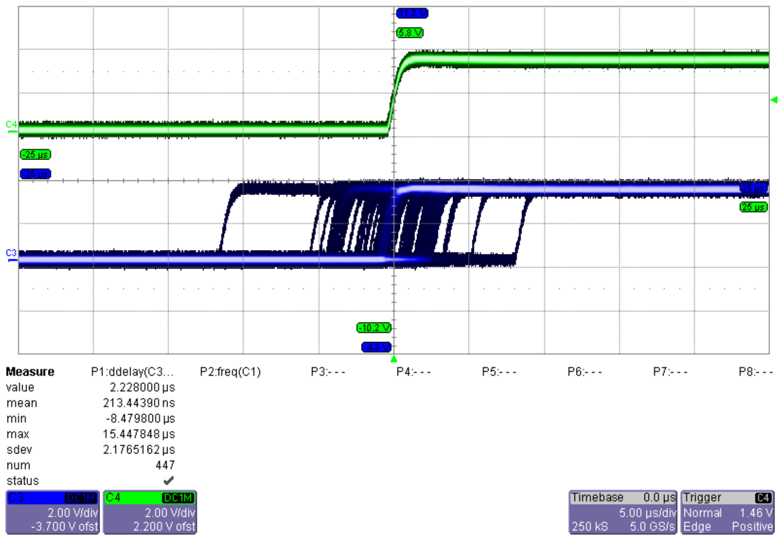
Figure 6.1: Four single measurements in normal mode.

6.3 Experiments

In this section we show the results of three different kinds of experiments. The first experiment shows TTE under normal operation. In the second experiment we will perform the above mentioned attack without our security layer. In this case, the attack will be undetected. The third experiment shows how our security layer can detect the attack.

6.3.1 Normal Mode of Operation

The goal of the experiment is to measure the average drift offset of two slaves in the normal mode of operation. In the Figure 6.2 a screenshot of the oscilloscope is shown. The upper signal represents the first slave node and is used as a trigger to provide a stable picture. The second signal represents the second slave and is recorded with a trace function allowing the changes of the drift offset to be observed. Each instance of the rising edge on lower signal represents one measurement and the horizontal difference between the two signals in that instant is the drift offset. In normal operation, the average drift offset is very small (in range of $\pm 1\mu s$). Figure 6.1 shows some of the measurements of Figure 6.2, but one measurement at the time.



```
Measure  P1:ddelay(C3...  P2:freq(C1)  P3:- - -  P4:- - -  P5:- - -  P6:- - -  P7:- - -  P8:- - -
value        2.228000 µs
mean       213.44390 ns
min         -8.479800 µs
max         15.447848 µs
sdev         2.1765162 µs
num                447
status               ✔
```

C3 DC1M	C4 DC1M	Timebase 0.0 µs	Trigger C4
2.00 V/div	2.00 V/div	5.00 µs/div	Normal 1.46 V
-3.700 V ofst	2.200 V ofst	250 kS 5.0 GS/s	Edge Positive

Figure 6.2: Two slaves in normal mode. The first slave is used as the trigger (green signal) and for every rising edge, the drift offset of the second slave is recorded (blue signal). Each trace of the blue signal represents one measurement and the horizontal difference the drift offset.

If the index of the measurements and the drift offsets are represented in a two-dimensional graph where the index of the measurements are on the X axis and the drift offset on Y axis, the graph has the shape of a straight line with occasional spikes. Figure 6.3 depicts the graph that shows the drift offset per measurement. The spikes in the drift offset values are due to delays created by the hardware, the driver software and the operating system.

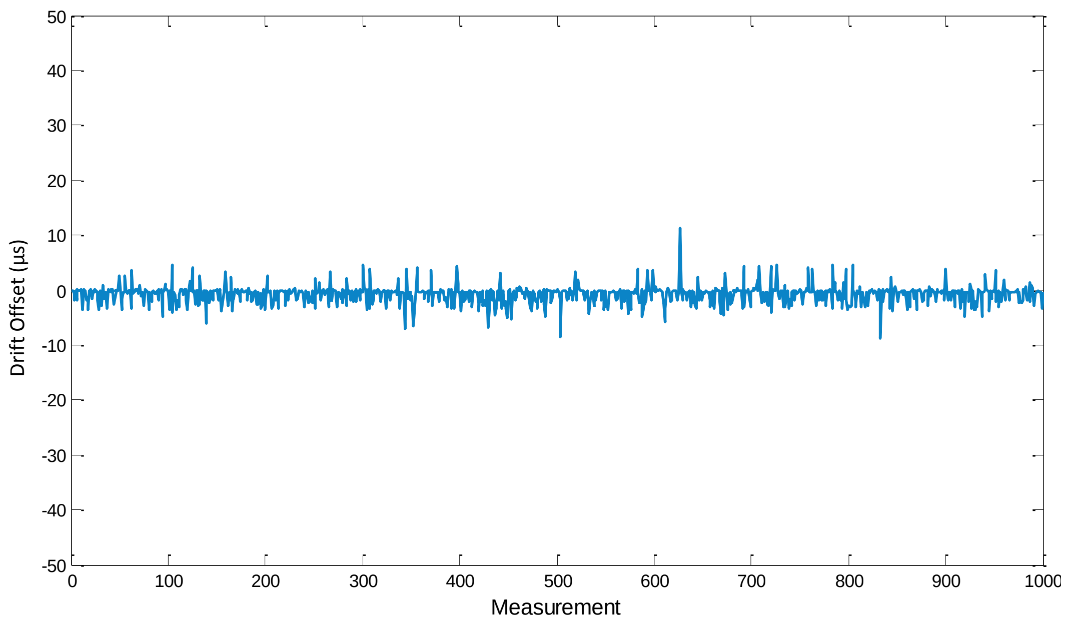
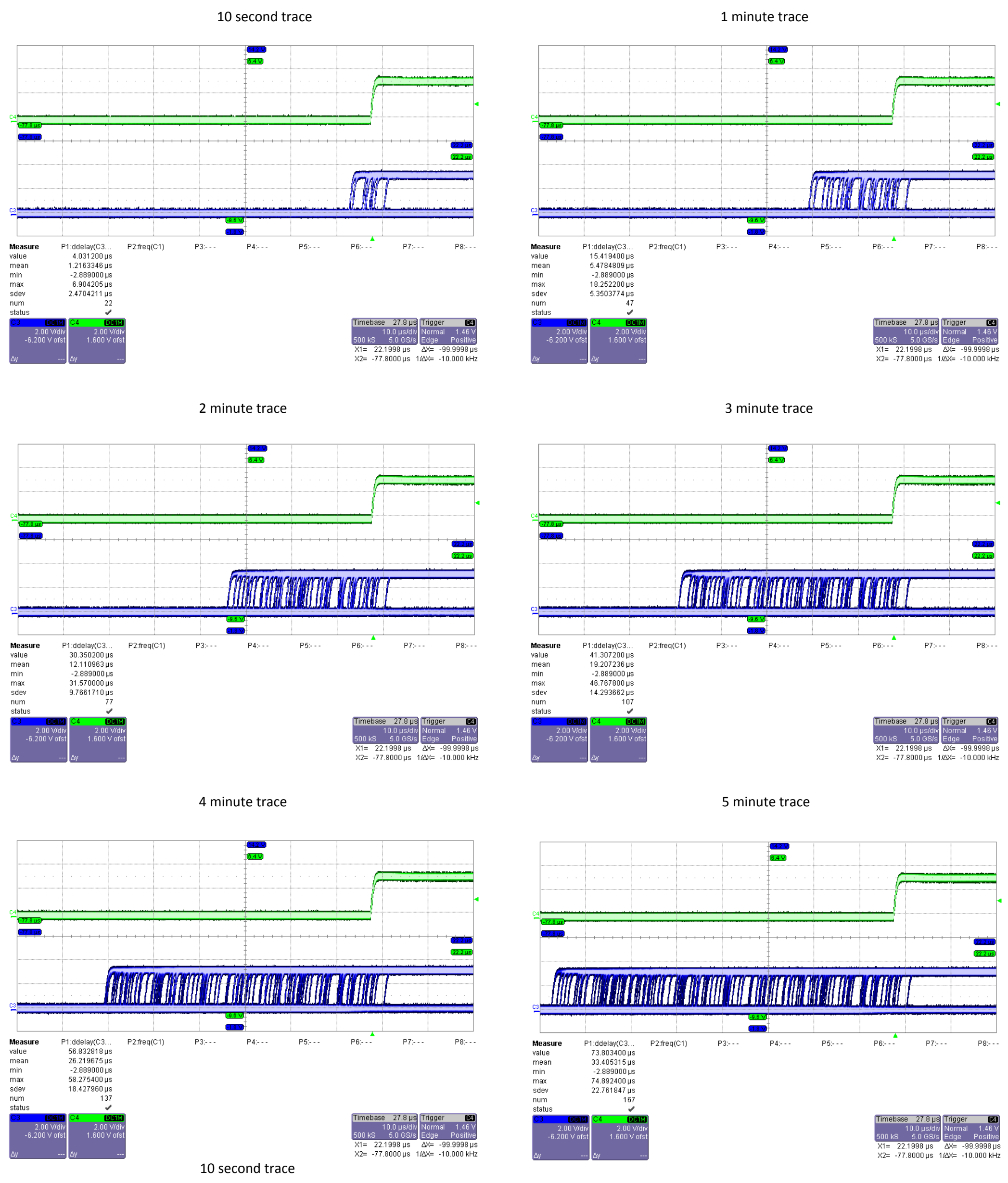


Figure 6.3: Drift offset of two slaves in the normal mode of operation represented as a trend function over thousand measurements

6.3.2 Attack Performed Without Security Layer

In this experiment the attacker modifies the timing of all messages received by Slave 2 (blue line in Figure 6.4). It gradually speeds up the synchronization messages received by Slave 2, staying always in the bound of the precision window. In each round, Slave 2 cannot detect the speed up, and adapts its local clock according to the speeded up clock synchronizing message. Speeding up the synchronization message makes Slave 2 adjusting its local clock to a future point in time. Since Slave 2 adjusts repeatedly its local clock to a future point in time, it will progressively develop a positive offset to the other nodes in the system. Figure 6.4 shows how the signals of Slave 2 arrive earlier and earlier compared to Slave 1 as time evolves (over a period of 5 minutes).

Figure 6.5 shows the growth of the drift offset over a longer period of time. Although the global time has drifted over 80 μs neither the TTE Controller nor the application has recognized the attack and they continued to operate under the perception that the present view on the global time is correct. If the functionality of the application depends on a reliable global time on all nodes, the system would fail.


10 second trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     4.031200 µs
mean      1.2163346 µs
min       -2.889000 µs
max       6.904205 µs
sdev      2.4704211 µs
num       22
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

1 minute trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     15.419400 µs
mean      5.4784809 µs
min       -2.889000 µs
max       18.252200 µs
sdev      5.3503774 µs
num       47
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

2 minute trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     30.350200 µs
mean      12.110963 µs
min       -2.889000 µs
max       31.570000 µs
sdev      9.7661710 µs
num       77
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

3 minute trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     41.307200 µs
mean      19.207236 µs
min       -2.889000 µs
max       46.767800 µs
sdev      14.293662 µs
num       107
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

4 minute trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     56.832818 µs
mean      26.219675 µs
min       -2.889000 µs
max       58.275400 µs
sdev      18.427960 µs
num       137
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

5 minute trace

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value     73.803400 µs
mean      33.405315 µs
min       -2.889000 µs
max       74.892400 µs
sdev      22.761847 µs
num       167
status    ✓
C3 DC1M  2.00 V/div  -6.200 V ofst  Δy ---
C4 DC1M  2.00 V/div  1.600 V ofst   Δy ---
Timebase 27.8 µs  10.0 µs/div  500 kS  5.0 GS/s
Trigger C4  Normal  1.46 V  Edge  Positive
X1=  22.1998 µs    ΔX=  -99.9998 µs
X2= -77.8000 µs  1/ΔX= -10.000 kHz
```

10 second trace

Figure 6.4: Oscilloscope trace of an undetected attack (over a period of 5 minutes)

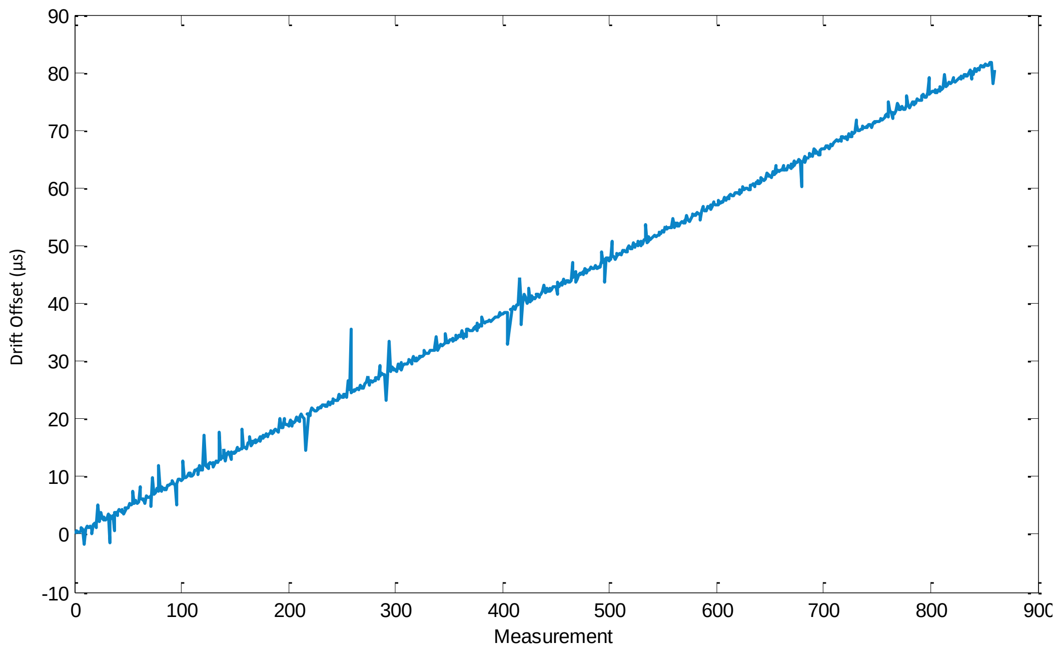
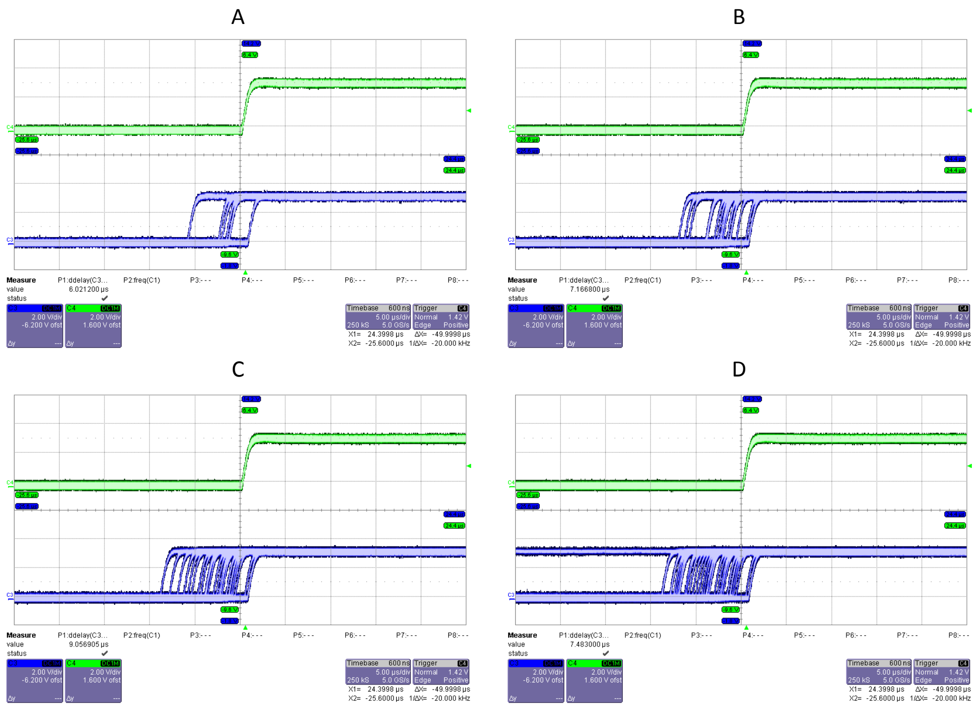


Figure 6.5: Trend function of the drift offset showing an undetected attack

6.3.3 Attack performed with active security layer

This experiment shows the attack of the last experiment, but with our activated security layer. In this experiment, the security layer successfully detected the attack and informed the application that the local view on the global time is no more trusted. The application can now decide how to act after having received this information. Typically, the application would go to a safe state. In our case, the application simple stops the TTE controller, and no more signal is generated (see Figure 6.6 and 6.7).



A

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value       6.021200 µs
status               ✔
C3  DC1M          C4  DC1M
    2.00 V/div        2.00 V/div
 -6.200 V ofst     1.600 V ofst
Δy          ---   Δy          ---
Timebase  600 ns   Trigger    C4
       5.00 µs/div Normal   1.42 V
250 kS   5.0 GS/s  Edge   Positive
X1=  24.3998 µs    ΔX=  -49.9998 µs
X2= -25.6000 µs  1/ΔX=  -20.000 kHz
```

B

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value       7.166800 µs
status               ✔
C3  DC1M          C4  DC1M
    2.00 V/div        2.00 V/div
 -6.200 V ofst     1.600 V ofst
Δy          ---   Δy          ---
Timebase  600 ns   Trigger    C4
       5.00 µs/div Normal   1.42 V
250 kS   5.0 GS/s  Edge   Positive
X1=  24.3998 µs    ΔX=  -49.9998 µs
X2= -25.6000 µs  1/ΔX=  -20.000 kHz
```

C

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value       9.056905 µs
status               ✔
C3  DC1M          C4  DC1M
    2.00 V/div        2.00 V/div
 -6.200 V ofst     1.600 V ofst
Δy          ---   Δy          ---
Timebase  600 ns   Trigger    C4
       5.00 µs/div Normal   1.42 V
250 kS   5.0 GS/s  Edge   Positive
X1=  24.3998 µs    ΔX=  -49.9998 µs
X2= -25.6000 µs  1/ΔX=  -20.000 kHz
```

D

```
Measure   P1:ddelay(C3...   P2:freq(C1)   P3:- - -   P4:- - -   P5:- - -   P6:- - -   P7:- - -   P8:- - -
value       7.483000 µs
status               ✔
C3  DC1M          C4  DC1M
    2.00 V/div        2.00 V/div
 -6.200 V ofst     1.600 V ofst
Δy          ---   Δy          ---
Timebase  600 ns   Trigger    C4
       5.00 µs/div Normal   1.42 V
250 kS   5.0 GS/s  Edge   Positive
X1=  24.3998 µs    ΔX=  -49.9998 µs
X2= -25.6000 µs  1/ΔX=  -20.000 kHz
```

Figure 6.6: Oscilloscope screenshot showing that the undetected drift is bounded by the active authentication protocol. After detection of the attack, the TTE controller of Slave 2 is switched of and no more trace is generated

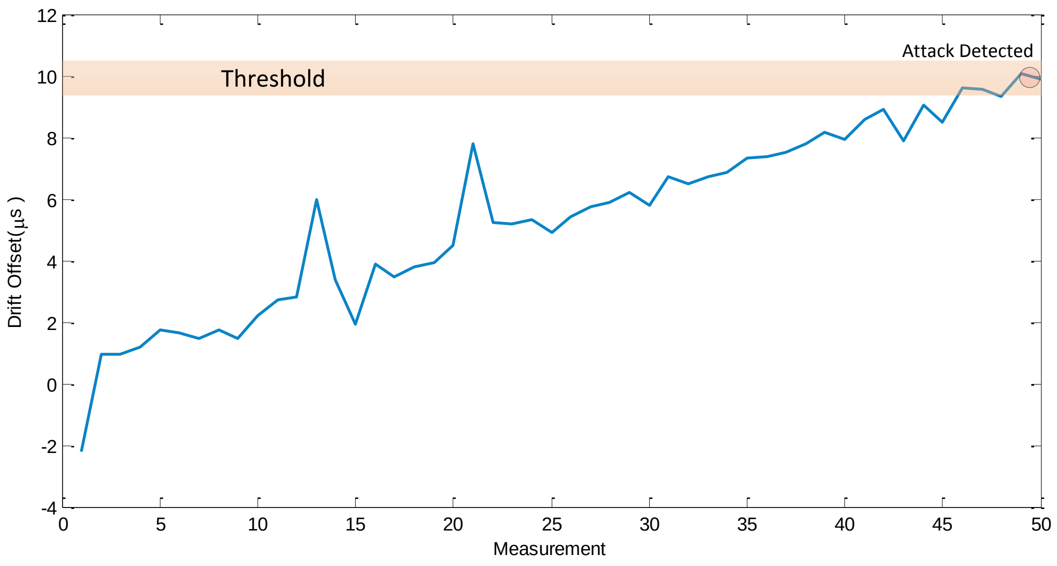


Figure 6.7: The attack is detected after the drift offset has exceeded the threshold

CHAPTER 7

Conclusion

This thesis has identified several major security threats against the global time in time-triggered systems and developed a security model to prevent various malicious attacks against the global time base. The security model is based on a secure authentication protocol designed to periodically check the global time in the network and report any anomalies. The authentication protocol operates according to the master slave principle where the master has the role of a trusted authentication authority. As a prove of concept, the authentication protocol is implemented on a Time Triggered Ethernet system as a security layer on top of the existing clock synchronization algorithm. The implementation comprises a set of hardware components, operating systems, application software and evaluation tools integrated in a single development environment.

We have experimentally demonstrated that without adequate additional protection, a TTE node can be unnoticeably brought out of synchrony by a malicious attacker. On the other hand, the security layer proposed in this work detects many different kinds of attacks, including the fabrication, modification, replay, delay or speed up of clock synchronization messages. The feasibility and efficiency of the proposed approach has been demonstrated by various experiments with simulated attacks. The implementation of the security layer was able to detected all attacks as specified by the attacker model.

Because of the obvious benefits, our approach has already been included in large-scale scientific and industrial projects like the ARTEMIS ACROSS [1] project lead by the Real-Time Systems Group of the Institute of Computer Engineering, at the Technical University of Vienna. In this project, the proposed approach is used to secure a global time base on multiple system-on-chip (SoC) components using a time triggered network on chip as a communication infrastructure. Although the architecture is fundamentally different from the implementation in this thesis, the security model proposed in this thesis withstands all challenges and provides promising results.

Concluding we can state, that the approach described in this thesis is independent from the actual implementation of the underlying platform and can be considered as a general model for securing the global time base in time triggered architectures.

[1] `http://www.across-project.eu/`

Bibliography

[1] Hermann Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1997.

[2] H. Kopetz, “The time-triggered architecture”, in *Proceedings of the First International Symposium on Object-Oriented Real-Time Distributed Computing*, Kyoto, Japan, April 1998, pp. 22–29.

[3] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer, “The time-triggered ethernet (tte) design”, *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Seattle, Washington*, May. 2005.

[4] “Ieee standard for a precision clock synchronization protocol for networked measurement and control systems”, *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. c1 –269, 24 2008.

[5] Hui Li, Yanfei Zheng, Mi Wen, and Kefei Chen, “A secure time synchronization protocol for sensor network”, in *Emerging Technologies in Knowledge Discovery and Data Mining*, Takashi Washio, Zhi-Hua Zhou, Joshua Huang, Xiaohua Hu, Jinyan Li, Chao Xie, Jieyue He, Deqing Zou, Kuan-Ching Li, and Mário Freire, Eds., vol. 4819 of *Lecture Notes in Computer Science*, pp. 515–526. Springer Berlin / Heidelberg, 2007.

[6] Armin Wasicek, “Security in time-triggered systems”, PhD.

[7] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song, “The tesla broadcast authentication protocol”, 2002.

[8] Causevic Emir, “A secure group communication middleware for time-triggered systems”, Master Thesis.

[9] Parameswaran Ramanathan, Kang G. Shin, and Ricky W. Butler, “Fault-tolerant clock synchronization in distributed systems”, *Computer*, vol. 23, pp. 33–42, October 1990.

[10] Andrew S. Tanenbaum and Maarten van Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[11] N.F. Ramsey, “The past, present, and future of atomic time and frequency”, *Proceedings of the IEEE*, vol. 79, no. 7, pp. 921 –926, July 1991.

[12] R. A. Nelson, D. D. Mccarthy, S. Malys, J. Levine, B. Guinot, H. F. Fliegel, R. L. Beard, and T. R. Bartholomew, “The leap second: its history and possible future metrologia”.

[13] D.L. Mills, “Internet time synchronization: the network time protocol”, *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482 –1493, October 1991.

[14] Ahmed El-Rabbany, *Introduction to GPS: the Global Positioning System*, ARTECH HOUSE, INC., Norwood, MA, USA, 2nd edition, 2002.

[15] S. Aslam-Mir, W. Haidinger, W. Elmenreich, T. Losert, and H. Kopetz, “OMG Smart Transducers, v1.0”, Smart Transducers Specification, 2003.

[16] John C. Eidson, *Measurement, Control, and Communication Using IEEE 1588 (Advances in Industrial Control)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[17] “Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications amendment 4: Media access control parameters, physical layers and management parameters for 40 gb/s and 100 gb/s operation”, *IEEE Std 802.3ba-2010 (Amendment to IEEE Standard 802.3-2008)*, pp. 1 –457, 22 2010.

[18] Petr Grillinger Hermann Kopetz, Astrit Ademaj and Klaus Steinhammer, “Time-Triggered Ethernet Protocol Specifcation”, Technische Universität Wien,Institut für Technische Informatik, Treitlstr.1-3/182-1, 1040 Vienna, Austria,, 2006.

[19] IEEE Standard Association IEEE EtherType Field Registration Authority., “EtherType Field Public Assignments.”, `http://standards.ieee.org/develop/regauth/ethertype/eth.txt`, 2005, [Online; accessed 30-May-2011].

[20] Klaus Steinhammer, *Design of an FPGA-Based Time-Triggered Ethernet System*, PhD thesis, Technischen Universität Wien, Dezember 2006.

[21] William Stallings and Lawrie. Brown, *Computer security : principles and practice / Willam Stallings, Lawrie Brown with contributions by Mick Bauer and Michael Howard*, Prentice Hall, Upper Saddle River, NJ :, 2008.

[22] Nancy G. Leveson, *Safeware: system safety and computers*, ACM, New York, NY, USA, 1995.

[23] Bruce Schneier, *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*, John Wiley & Sons, Inc., New York, NY, USA, 1995.

[24] Whitfield Diffie and Martin E. Hellman, “New directions in cryptography”, 1976.

[25] B. Harris, “Rsa key exchange for the secure shell (ssh)”, `http://tools.ietf.org/html/rfc4432`, March 2006.

[26] E. Rescorla T. Dierks, “The transport layer security (tls) protocol”, `http://tools.ietf.org/rfcmarkup/5246`, August 2008.

[27] Victor S Miller, “Use of elliptic curves in cryptography”, in *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, New York, NY, USA, 1986, pp. 417–426, Springer-Verlag New York, Inc.

[28] Neal Koblitz, Alfred Menezes, and Scott Vanstone, “The state of elliptic curve cryptography”, *Des. Codes Cryptography*, vol. 19, pp. 173–193, March 2000.

[29] T.S. Denis and S. Johnson, *Cryptography for developers*, Safari Books Online. Syngress Publishing, Inc., 2007.

[30] National Institute of Standards and Technology, “Digital signature standard (dss)”, 2000.

[31] National Institute of Standards and Technology, “Digital signature standard (dss)”, 2009.

[32] Soekris Engineering inc., “net4801 series boards and systems. user’s manual”, 2004.

[33] Inc. Advanced Micro Devices, “Amd geode^{TM} gx processors data book”, August 2005.

[34] Klaus Steinhammer and Astrit Ademaj, “Hardware implementation of the time-triggered ethernet controller.”, in *IESS*, Achim Rettberg, Mauro Cesar Zanella, Rainer Dömer, Andreas Gerstlauer, and Franz-Josef Rammig, Eds. 2007, vol. 231 of *IFIP*, pp. 325–338, Springer.

[35] Altera Corporation., “Altera Cyclone II”, `http://www.altera.com/products/devices/cyclone2/cy2-index.jsp`, [Online; accessed 12-July-2011].

[36] L. Dozio and P. Mantegazza, “Linux Real Time Application Interface (RTAI) in low cost high performance motion control”, *Motion Control*, pp. 27–28, 2003.

[37] Voyage Design and Consultants, “Voyage Linux”, `http://linux.voyage.hk/`, [Online; accessed 14-July-2011].

[38] Intel Corporation, “Preboot Execution Environment(PXE) Specification 2.1”, 1999.

[39] K. Sollins, “The tftp protocol (revision 2)”, `http://tools.ietf.org/html/rfc1350`, July 1992.

[40] D. Robinson R. Thurlow Sun Microsystems Inc. C. Beame Hummingbird Ltd. M. Eisler D. Noveck Network Appliance Inc. S. Shepler, B. Callaghan, “Network file system (nfs) version 4 protocol”, `http://tools.ietf.org/html/rfc3530`, April 2003.

[41] Eclipse Foundation, “Eclipse IDE”, `http://www.eclipse.org/`, [Online; accessed 14-July-2011].

[42] Tom St. Denis, “LibtomCrypt Website”, `http://libtom.org/?page=features&newsitems=5&whatfile=crypt`, [Online; accessed 14-July-2011].

[43] Tom St. Denis, “LibtomCrypt Developer Manual”.

[44] Tetsu Iwata and Kaoru Kurosawa, “Omac: One-key cbc mac”, in *Pre-proceedings of Fast Software Encryption, FSE 2003*. 2002, pp. 137–161, Springer-Verlag.

[45] M. Bellare, J. Kilian, and P. Rogaway, “The security of cipher block chaining”, 1994.

[46] National Institute for Standards and Technology, “Recommendation for block cipher modes of operation: The cmac mode for authentication”, NIST Special publication 800-38B, May 2005.