SECRET

SECurity of Railways against Electromagnetic aTtacks

Grant Agreement number: 285136
Funding Scheme: Collaborative project
Start date of the contract: 01/08/2012
Project website address: http://www.secret-project.eu

Deliverable D 4.3

Simulation and assessment results of dynamic protection system

Deliverable on simulation and assessment of dynamic Protection System
Date: 29/08/2014
Distribution: PU
Manager: Fraunhofer

Document details:

Title	Simulation and assessment of dynamic protection system
Work package	WP4
Date	22/07/2014
Author(s)	M. Kober (Fraunhofer), R. Klein (Fraunhofer)
Responsible Partner	Fraunhofer
Document Code	SEC-D4.3-C-082014-Simulation and assessment results of dynamic protection systems-FIAS-Final
Version	C
Status	Final

Dissemination level:

Project co-funded by the European Commission within the Seventh Framework Programme

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission) Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document history:

Revision	Date	Authors	Description
0.1	18/10/2013	Fraunhofer	Initial version of D4.3
0.9	04/06/2014	Fraunhofer	Model/Scenario description
0.95	17/07/2014	Fraunhofer	Simulation results
1.0	27/07/2014	Fraunhofer	Final version with minor revisions following internal review

Table of content

1 Executive summary

The objective of the attack-centered assessment of the dynamic protection system is to show that the attacks identified in the analysis carried out in WP1 are effectively coped with at the dynamic protection level. We simulate different attack scenarios in the communication infrastructure of railways. To do this we choose a suitable simulation method. At the beginning, the simulation describes real world railway infrastructures and in later enhancements we integrate new components of the new dynamic protection system. This dynamic protection system is under research by our partners and the conceptual specifications (in D 4.1 and D4.2) are the input sources for our work. So the simulation grows the more details we get. At the end we have a comprehensive simulation which includes the real world components and the virtual dynamic protection system specification. By analysing our simulation experiments we can provide concrete explanations about how this system would work under real conditions in a real environment.

2 Introduction

2.1 Purpose of the document

The purpose of the Deliverable 4.3 is the assessment of the initial dynamic protection system as defined within Tasks 4.1 and 4.2. The results will be a refined, possibly amended architecture infrastructure, resilient compound network specification and health/attack manager specification. In order to achieve this aim, a simulation based evaluation of the dynamic protection system (resilient communication architecture and health/attack manager) will be performed, using data farming experiments in order to generate statistic data about the effectiveness of the dynamic protection system in different attack scenarios. Due to the iterative work methodology proposed within the SECRET project, this task will in fact consist of multiple repetitive phases.

The document consists of 5 main sections. After the introductory chapters we describe the state of the art in multi-agent simulation systems in Chapter 3. Within the SECRET project we use Fraunhofer's modelling and simulation tool SyMo/Ready that is based on a multi-agent based simulation concept to be outlined in Chapter 4.

In the following Chapter 5 we describe the Secret dynamic detection system, the multipath communication system, and also the railway infrastructure as modelled and to be simulated in SyMo/Ready. We specify in which way these components can be modelled as SyMo agents and how the communication between them can be established to be simulated in Ready. The results of this analysis flow into our agent behaviour/attributes description and will be modelled within the simulation

To run simulations there are several use cases and scenarios available described in D4.1 [11] and D4.2 [12]. Chapter 6 will use them and adapt them to create simulation experiments. We specify what the components have to provide and present the results of the simulations.

Finally, in Chapter 7 of the document we summarize our results and draw the conclusions.

2.2 Definitions and acronyms

	Meaning
DS	Detection System
CHAM	Central Health/Attack Manager
HAM	Health/Attack Manager
RHAM	Railway Health/Attack Manager
THAM	Train (onboard) Health/Attack Manager
AS	Acquisition System
MBCS	Multipath Bidirectional Communication System
ERTMS	European Railway Traffic Management System
EM	Electromagnetic

3 State of the art

3.1 Why Models and Simulations in SECRET

Railway communication systems are already today complex technical systems with relations to humans, organisational structures, legal and regulatory frameworks, etc. In the near future, the technical systems will become even more complex and diverse including new technologies, new services, a greater variety and combination of systems and components, etc.
Secret's main purpose is to investigate how such complex systems behave under electromagnetic attacks of various kinds, how they can be protected and become resilient against them. A broad spectrum of measures, system architectures, and technologies are investigated with respect to this goal (see esp. the project's deliverables D4.1 [11] and D4.2 [12]).
Each of the technologies, systems components, system architectures relevant in this context can be described at various levels of detail – but the main question remains: how do all these systems and components play together under various conditions? What is the resulting *overall system behaviour*? This is by far not a trivial issue. To find out a system's behaviour means to get a detailed understanding of the behaviours of its subsystems and components and of their interactions. Temporal aspects, various physical effects, material properties, dependencies between functions, services, and processes, etc. need investigation.
For this purpose, WP4 of the Secret project includes complex system simulations. We investigate how different kinds of railway communication systems behave under electromagnetic attacks. We describe the behaviours of its components, their interplay in different system architectures, and different types of attack scenarios in order to investigate and evaluate the resulting overall system behaviour.
The question arises what the best way is to investigate such complex systems. We need models of subsystems and components which include all relevant physical and technical attributes and modes of behaviour as well as dependencies between them (resources they need and provide, services they deliver, etc.). Such models are built on an appropriate level of granularity which depends on the effects and behaviours of the overall system to be investigated.
Models are primarily "static" descriptions of systems. In order to understand their behaviours we have to get them into live: we need system simulations. Parts of such simulations are scenarios: appropriately specified sequences of environmental changes, their effects on systems and components, and maybe actions undertaken by control systems to react to them.
Today, there are many different kinds of models and simulations. In order to find out which of them best fit for our purposes we have to specify the main issues we need in our case:

- There is a broad spectrum of different kinds of systems and components to be taken into account in our models and simulations: [6]
- They all have different kinds of physical, technical, etc. properties to be taken into account ;
- They show well defined but different kinds of physical and technical behaviours under various conditions ;
- They depend on each other through resources and services they need and provide ;
- They can be configured in different system architectures resulting in different ways

of interactions ;
- Temporal aspects are an important issue in understanding and evaluating the overall system behaviour ;
- We need flexibly specified scenarios as well-defined sequences of changes, events, and actions.

In addition to these functional requirements to models and simulations there are various non-functional requirements resulting from practical considerations:

- The models must be intuitively understandable and transparent.
- They must be easy to built, to adapt and to modify.
- They should be modular in order to provide the necessary modeling and simulation flexibility.
- They have to support different levels of abstractions.
- Their simulations must be computationally manageable.

Multi-agent models and simulations have proven to be adequate for such complex model investigations (see also [6, 7, 8, 10] and references therein): they provide the necessary expressiveness combined with great flexibility.

3.2 Multi-Agent Based Models and Simulations

Multi-agent simulation is based on the formulation of the model as a multi-agent system, i.e. as a system composed of interacting, autonomous agents, in an environment that is also part of the model. There is a broad spectrum of agent-based models which ranges from relatively simple models (traffic simulations with pedestrians or cars) up to models which mimic human actors as intelligent agents (for detailed overviews see [6, 7, 8, 10]).

The main issues specified above for agent based models and simulations in Secret can be related to agent models as follows:

- different kinds of systems and components: each of them can be represented by a corresponding agent type ;
- with all relevant kinds of physical, technical, etc. properties;
- each agent type comes with all needed kinds of physical and technical behaviours where the various conditions needed for a certain behaviour are specified as part of the agent model;
- agents need and provide resources and services;
- different system architectures can be built by relating the various kinds of agents to each other in a concrete system configuration ; this may include different levels in a hierarchy;
- Temporal aspects can be specified as part of the agent model and be used in the overall simulations of the system behaviour ;
- Scenarios can be defined as sequences of events and actions applied to agents changing their internal states, triggering their appropriate behaviours and reactions, and propagating through the network of dependencies.

3.3 Fraunhofer System Modeler and Simulation Environment SyMo/Ready

Fraunhofer IAIS explores system models and their practical usage. Rational and targeted

acting needs a model of the system that is going to be affected. Without a system model there is no rational understanding about entities and their dependencies. This applies to both physical and derived technical and biological systems, as well as social, economic and business systems. A system model consists of an amount of components (agents) and an amount of behaviours of these components. At each time the individual components can take a specific state and show a specific behaviour defined by a set of conditions.

Possessing a system model opens an understanding of the structure of the modeled system. The model describes “what” the systems consists of and how these “what” relates to each other.
A system model has, of course, only a value when the predicted system states of the model also occur in the real system or when the difference between prediction and real system is acceptable. The predicted states can consist of scalar values or probability distributions.

This combination of explanation and prediction model is the core concept of all cognitive model-based natural sciences.

4 Fraunhofer System Modeler and simulation environment SyMo/Ready

Our agent based modelling and simulation environment SyMo/Ready (see [1], IRRIIS [2], DIESIS [3], EMILI [4, 5]) used in Secret can be described as a *light weight agent based* system: the agents are persistent software entities with their own types, internal data, and explicitly defined behaviours (procedures). They communicate with each other in pre-defined ways and execute pre-defined behaviours – resulting in complex system behaviour simulations. SyMo agents are especially designed to simulate *technical* system behaviours of various kinds and complexity. They do not support complex *reasoning* as cognitive agents typically do maintaining their own sets of beliefs, desires, and intentions, nor are they so simple that they are just data containers (as, for instance, in some traffic or logistics simulations).
SyMo/Ready consists of two main components:

- The System Modeler SyMo: a proprietary Java based modeling software. It allows us to model any kind of complex system as a collection of interacting agents. Such models include all relevant subsystems, components, their attributes, behaviours, and dependencies. Attack scenarios can be specified as part of SyMo models allowing us to model arbitrary sequences of external and internal events, control reactions, propagations along dependency chains, etc.

- The simulation system Ready is tightly coupled with SyMo and able to interpret SyMo models according to the intended semantics of behaviours and scenarios. The SyMo model includes the entire information about all interacting agents, their behaviours/attributes and also the experiment description about the kind of electromagnetic attack scenario. But there is no explicit logic in SyMo that processes anything. SyMo models only describe what to do, which data is necessary and define the command chain of a simulation run. Ready provides an implementation frame in JAVA where you can fill the modelled behaviours with logic and calculation methods according to the simulation requirements. Ready also provides the execution sequence

to import the SyMo model file, to instantiate the model content, to run a concrete simulation and to generate the output file.

The SyMo/ready agent environment allows us to model the static railway communication infrastructures as well as the dynamically changing processes inside the communication systems.
In order to illustrate the functionality and modeling process in SyMo in detail we will now use the implementation of the mentioned railway infrastructure and communication system including the new security system. Since the new security system, called Dynamic Protection System (DPS), is closely linked to the railway infrastructure topology and the existing communication system, both system are included into the DPS model. It is not necessary to model all systems in parallel.

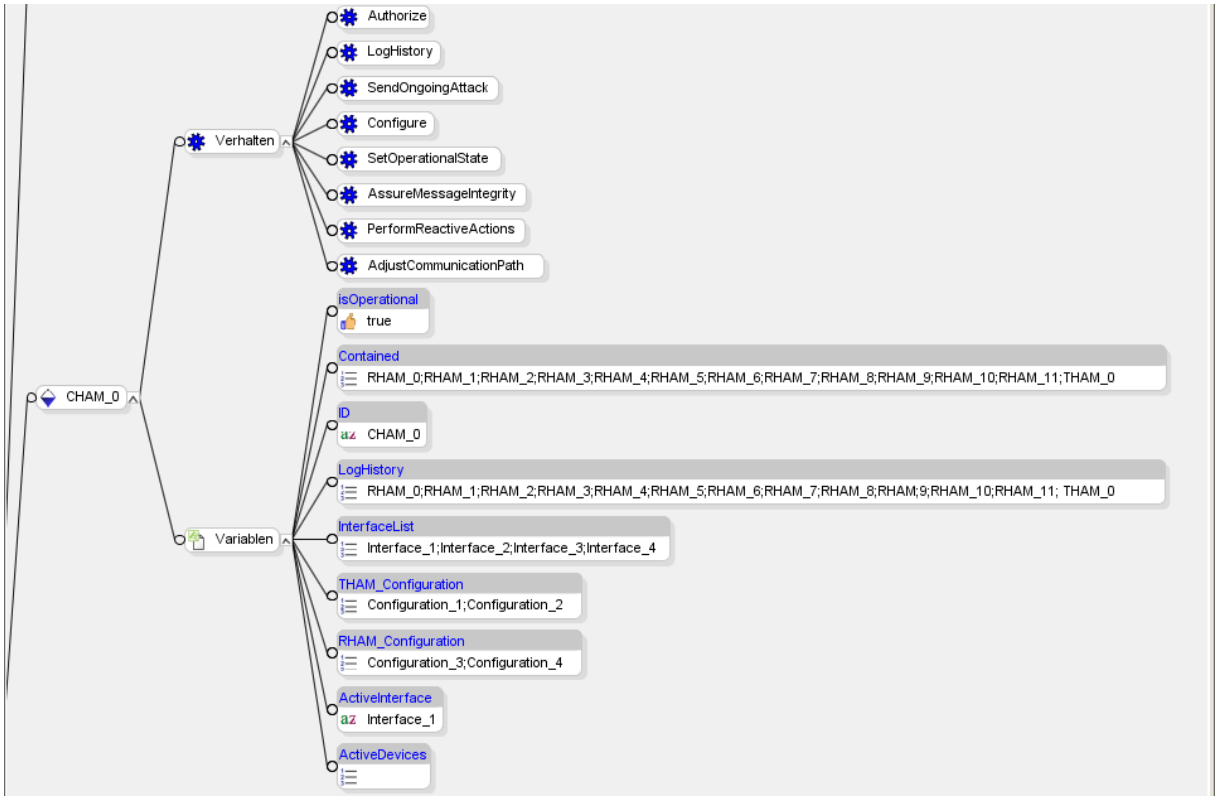


Fig. 1 Example model of a SyMo agent: the Central Health/Attack Manager (CHAM_0). It contains several behaviours ("Verhalten") and a lot of attributes ("Variablen")

Each component or subsystem in a Secret model is described as a single agent. Fig. 1 shows as an example a schema of the Central Health/Attack Manager CHAM as one component in the new Secret Dynamic Protection System. It contains a set of variables with all relevant "proprietary" data ("Variablen" see below) and behaviours ("Verhalten" see below) of such an agent. These are the two main components of each agent.

Agents also get a specific type that describes its position in the components hierarchy. This means, that there are components with a higher position in the communication hierarchy and others with a lower one. We define a components hierarchy (figure 2: Agent hierarchy legend) to describe which agent is able to interact with which other agent. On the other side this definition avoids that agents interact with (in reality) not accessible agents. The Central

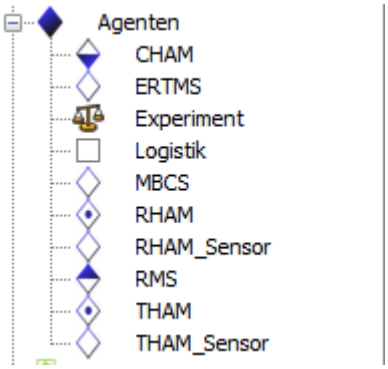


Fig. 2: Agent hierarchy legend

Health/Attack Manager occupies a high position in the hierarchy because it has as the name suggests a lot of central tasks in the Dynamic Protection System.

Now we focus on the two agents' main components.

- **Attributes (variables):**

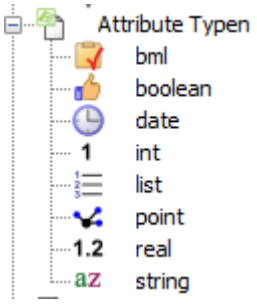


Fig. 3: Agent attributes legend which defines all needed parameter types

All behaviours work with attributes (fig. 4) which describe the internal data of an agent. They can only be accessed and changed by behaviours defined for this agent. They define the starting data for a simulation run, and they will be modified during a simulation run by the execution of agent behaviours. Each attribute gets a specific type (figure 3: Agent attributes legend) that defines the kind of the attribute like integer values, lists, strings. For example, there is one attribute "isOperational". This Boolean parameter describes the working state. At the beginning of a simulation run this attribute is set to the value "true". The agent is working and can start processing behaviours. During simulation runtime a situation may occur where it is necessary to disable the agent. For example, if the behaviour "SetOperationalState" is triggered by another agent event the Central Health/Attack Manager is forced to shut down its accessibility for other agents. This happens by setting the "isOperational" attribute to false.

- The isOperational state, modeled as a Boolean value, that defines the mode of the agent if it is running or not

- The unique ID, modeled as a String value, to identify a known agent in different parameter checks. By using the unique ID the CHAM can check the components validity. A wrong or invalid ID will be logged and interpreted as possible compromising attack into the communication network.

- The initial activeInterface configuration, modeled as a String value, which defines the used communication interface at the beginning of a simulation run.

- **Behaviours:**

And an agent contains of a set of behaviours which are implemented as atomic tasks. During the start sequence of one simulation run which consists of hundreds of single simulation cycles the agents are instantiated with the modeled start configuration defined in SyMo, e.g.:

- The “Authorize” behaviour verifies the authorization request of a new agent. At simulation start all the CHAM underlying agents have to register at the CHAM that they are available and active. The CHAM checks their unique ID. By this the CHAM can decide if a new agent is a valid one or maybe is compromised by an attack event. The authorization behaviour is also used in a similar way during the continuous “KeepAlive” signal of each active agent.

- After a successful registration process the “Configure” behaviour provides each underlying agent a specific configuration setup with information about what to do in the case of an attack and some other information.

- If an agent registration process fails by any reason the CHAM has the option to disable the affected agent using the “SetOperationalState” behaviour.

- If the communication between train and CHAM is lost the “AdjustCommunicationPath” behaviour changes the “activeInterface” configuration. This is to re-establish the communication via a new communication interface (e.g. from GPRS to LTE frequency).

- The “LogHistory” behaviour is always running and logs every event, behaviour process and state change of all agents including the timestamps when they occur. This is necessary for later analysis of simulated scenarios and attack situations.The other main component of SyMo agents are behaviours:

So, for each agent type we can define one or several behaviours. Behaviours are procedures uniquely assigned to an agent which strictly use the agent’s internal attribute data. The interaction capabilities of an agent result from the execution of these behaviours. A behaviour describes what an agent can do depending on specific trigger events that initiate the process of a behaviour during simulation runtime. Behaviours include sending messages to related agents initiating there adequate behaviours – and so forth.

Trigger events are:

- A new Health/Attack Manager agent registers at CHAM in the concrete simulation run for the first time.

- One or more Health/Attack Manager agents anywhere in the railway network topology detect an ongoing electromagnetic attack.

- The communication signal between train and Central Health/Attack manager agent is lost

There are a lot of more crucial trigger events like those mentioned and they all need a special treatment within the simulation. E.g. a detected electromagnetic attack initiates cascading behaviour performance in different agents because of the high impact of an attack.

We distinguish between active and working behaviours. A behaviour is always active but not necessarily working during simulation runtime on condition that the corresponding agent is also active. This is absolutely necessary because a behaviour is only executed when certain trigger parameters apply. For that it must be active. One could call the active state of a behaviour as a kind if idle mode. It is waiting for trigger events which initiate the work process. And these trigger events are one part of each behaviour. The implementation of a behaviour consists of several basic parts. The pre-conditions: a set of logical expressions on an agent's attribute data which must be fulfilled in order to trigger the according behaviour (bringing it into operational state 'working'); the execution part: a procedure to be executed if the behaviour is in state 'working'.

Implementation of a behaviour in Ready

1. Pre-conditions review of relevant input parameters as trigger events. Trigger events provide information about the agents' internal state according to specific rules or are defined as an incoming message event from another connected agent triggering the accents specific reaction.. The verification of these events and the analysis of the input parameters start the behaviour process at all. The following boxes show some examples of typical Secret agent preconditions which have to be checked for behaviour execution.

   - Simulation state: Is the simulation still running? If the simulation is not running or has been stopped by an e.g. exception or any other runtime condition a behaviour mustn't be performed.

```
//get simulation state
BoolVal einsatzRunning = (BoolVal)State.Get (GetAgent
(globalConst.GET_STRING_COMMUNICATIONTRAFFIC()).FullPath
(globalConst.GET_STRING_IS_RUNNING())).GetValue ();

//if simulation is not running then interrupt behaviorbehaviour process for
this cycle
if(!einsatzRunning.Get ()) return new ConfidenceFinished (this, Now, new
Explanation (GetName (), new ExplanationItem (this,
globalConst.GET_STRING_COMMUNICATIONTRAFFIC_ZU_ENDE())));
```

   - Agent state: Is the agent still active? If in case of a detected electromagnetic attack one specific agent, e.g. a railway Health/Attack Manager agent had to be disabled by the Central Health/Attack Manager then this agent should no longer send data signals.

```
//get agents' operational state
BoolVal isOperational = (BoolVal)State.Get (
FullPath ("isOperational")
).GetValue ();
```

- Attack situation: Is the agent currently under attack? If an agent detects an electromagnetic attack via his sensor network it immediately has to alert the Central Health/Attack Manager about the new attack situation.

```
//get agents' attack state
BoolVal underAttack = (BoolVal)State.Get (
FullPath ("underAttack")
).GetValue ();
```

We now consider the concrete implementation of the attack agent and its behaviour trigger events that are reviewed by preconditions test.

```
//get simulation state
BoolVal einsatzRunning = (BoolVal)State.Get (GetAgent
(globalConst.GET_STRING_COMMUNICATIONTRAFFIC()).FullPath
//if simulation is not running then interrupt behaviorbehaviour process for
this cycle
if(!einsatzRunning.Get ()) return new ConfidenceFinished (this, Now, new
Explanation (GetName (), new ExplanationItem (this,
globalConst.GET_STRING_COMMUNICATIONTRAFFIC_ZU_ENDE())));
 (globalConst.GET_STRING_IS_RUNNING())).GetValue ();


//get agents' operational state
BoolVal isOperational = (BoolVal)State.Get (
FullPath ("isOperational")
).GetValue ();
//if attack agent not running then interrupt behaviorbehaviour process for
this cycle
if(!isOperational.Get ())
{return new ConfidenceRunning (this, Confidence.PlanRunning, 0, 1.0, Now, new
Explanation (GetName (), new ExplanationItem (this, "Attack Device
inactive")))}
//if attack agent is running then enter main execution body
else{
/**
 * Main execution body
 */
```

2. Main execution body with behaviour tasks. Fig. 6 illustrates the execution part of one agent behaviour with the concrete implementation of what the agent has to do if the pre-conditions are fulfilled.

```
//if attack agent is running then enter main execution body
else{
/**
 * Main execution body
 */
```

Here we implement the concrete agent specification. The algorithm is defined including which information will be imported from other agents like calculation attributes, timer, etc. E.g. the attack device agent has one behaviour called „Attack“. After the trigger parameters are positively checked the main implementation code will be performed. It defines the agents' task in this behaviour: Attack. The agent gets all information that describes the behaviour specification.

- The targeted component of the railway network. E.g. in this simulation run the attack device scrambles railway Health/Attack Manager with the ID „RHAM_5“

```
//get attacked device
AgentListVal attackedComponent = (AgentListVal)State.Get (
FullPath ("AttackedComponent")
).GetValue ();
```

- The attack impact value that defines how strong or maybe weak the attack will be performed. E.g. a low impact value is defined as weak disturbing signal in the communication network. Communication is still running. The attacked component can detect the attack and report it to the Central Health Attack Manager. The communication network is alerted about the ongoing attack but no countermeasures are necessary as far as the communication is not lost. As opposed to this a high impact will strongly disturb and interrupt the communication in its scope. The attacked device is indeed able to detect the attack but because of the communication loss the attacked component is no longer able to report the attack to the Central Health/Attack Manager agent

```
//get impact value
StringVal impactValue = (StringVal) State.Get (
FullPath ("ImpactValue")
).GetValue ();
```

attack duration is responsible for the timeframe in which the attack device is actively sending transients.

```
//get attack duration
IntVal attackDuration = (IntVal) State.Get (
FullPath ("AttackDuration")
).GetValue ();
```

- The attack count defines the repetition rate of the attack device.

```
//get attack count
IntVal attackCount = (IntVal) State.Get (
FullPath ("AttackCount")
).GetValue ();
```

- E
ach agent contains a Boolean flag value that enables or disables the attack situation. A true value indicates that the agent is currently under attack. On the other side a false value tells that the agent is properly working within

```
//get underAttack flag value of going to be attacked agent
BoolVal underAttack = (BoolVal) State.Get (GetAgent (attackedComponent.Get
(0).GetName ()).FullPath ("underAttack")).GetValue ();
```

normal parameters.

- The attack will now be performed by processing the mentioned parameters

```
//attack agent logic
if (!underAttack.Get() && attackCount.Get () > 0)
{
	try
	{
		//set underAttack parameter of attacked agent to „true"
		Actions.Put (new Tag (GetAgent (attackedComponent.Get (0).GetName
		()).FullPath ("underAttack"), new BoolVal (true)));
		//reduce the attackCount minus 1
		Actions.Put (new Tag (FullPath ("AttackCount"), new IntVal
		(attackCount.Get()-1)));

		System.out.println(System.currentTimeMillis () + " Attack in
		progress: " + attackedComponent.Get (0).GetName () + " with " +
		impactValue.Get () + " Impact" );
	}
	catch (Exception e)
	{
		//exception handling if any of the actions fails
		e.printStackTrace();

	}
}
```

The attack is performed in the main body. Among other thing, you can see the algorithm is going to change the “underAttack” parameter into “true”. Before that the value was originally “false” at simulation start. Here it is very important to know that this specific state change will be performed during the actual simulation cycle but it will be finally acknowledged when the attack behaviour process and the simulation cycle is finished. The new agent state is set as start configuration for the next simulation cycle.

3. The last part of a behaviour is about changing states or modifying attributes for the next simulation cycle. In case of an electromagnetic attack the Central Health/Attack Manager may set the state of another agent into disabled mode and for the next simulation cycle the inactive agent will no longer be available for the complete simulation cycle.

```
//finalize ~~behavior~~behaviour execution and acknowledge all changed parameters
//so the simulation can plan the next simulation cycle
return new ConfidenceRunning (this, Confidence.PlanRunning, 0, 1.0, Now, new
Explanation (GetName (), new ExplanationItem (this, "Attack runs")));
```

Scenario description

One simulation run which is performed by the Ready simulation environment is modeled as a SyMo experiment. Here we define that a SyMo experiment model and a SECRET scenario description are used as synonyms. A SECRET scenario description is the literally formulated specification of an attack event written within the SECRET preparations. Fraunhofer translates the SECRET scenario description into the SyMo simulation experiment. Next to agent models a SyMo model describes the experiment that will be performed. The experiment uses a special syntax (see Fig. 4).

Fig. 4: Experiment model which is necessary to initiate the simulation process

An experiment requires four components which together are necessary for the simulation performance: behaviour (“Verhalten”), Main execution, Breed and attributes (“Variablen”). There is only one behaviour for an experiment called “EXECUTE” (fig. 5).

Fig. 5: The EXECUTE behaviour starts the simulation process

The “EXECUTE” behaviour is the most important one for any kind of simulation in Ready. It is responsible for all commands that have to be performed in the main execution part. While Ready is running and interpreting the imported SyMo model this behaviour checks if all modeled agent parameters and behaviours are syntactically consistent with their implementation equivalent. Remember, the SyMo model only gives a behaviour a designated name but there is no logic or algorithm beyond that. The logic is implemented in the corresponding JAVA class in Ready. The EXECUTE behaviour in Ready maps

SyMo behaviour and the appropriate behaviour implementation. In the next step the "EXECUTE" behaviour plans the complete simulation run.
For this purpose there is the main execution part. The main part encloses the simulation initialization and the order of parallel and sequential behaviour executions.

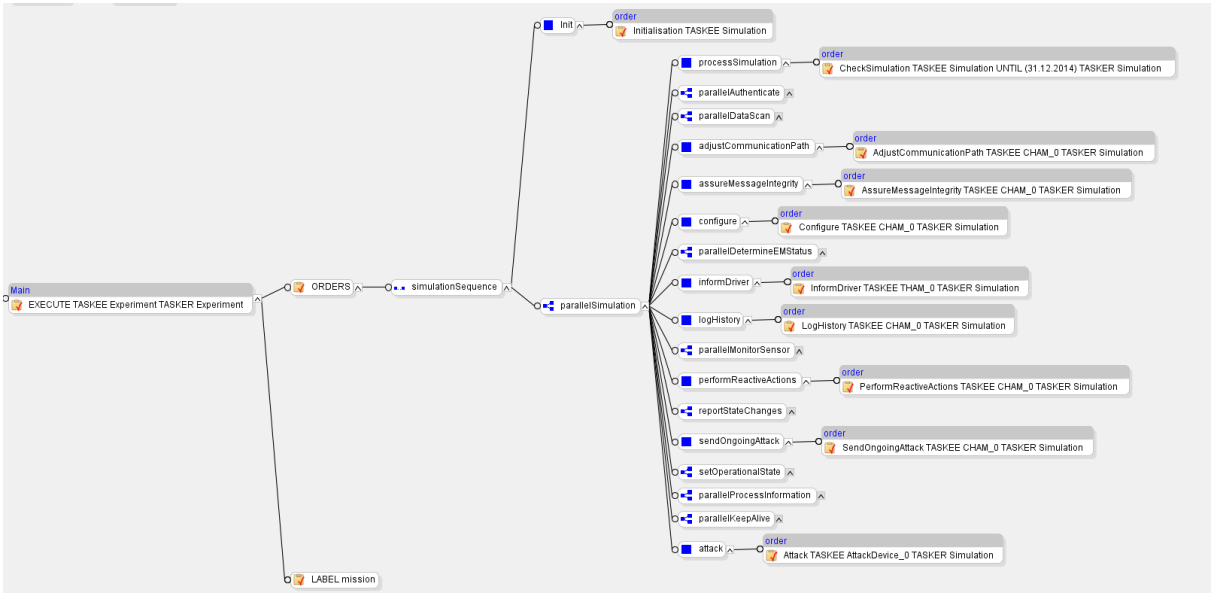


Fig. 6 Process of one single simulation cycle. It describes the simulation process where all behaviours are categorized and specified as parallel or sequential processes.

In SyMo one can create various orders of parallel and sequential processes. There is no limit in modeling in this respect. The challenge is to define which behaviours have to be executed in parallel and which in a sequence. SyMo provides some standards to define orders. They are listed in figure 6. To define an execution tree in SECRET we use the process order types "Elementary", "Parallel" and "Sequence".

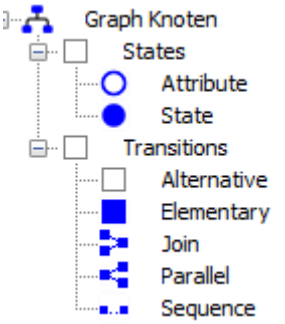


Fig. 7: Order legend which defines the behaviours as e.g. parallel or as a sequence

Elementary commands are those that are performed only in specific cases and which are associated with a single and uniquely specified agent. For instance, the "adjustCommunicationPath" behaviour is an elementary command because it is executed when several parameters simultaneously apply in a specific case for one agent. This case occurs when the attack event runs, the attack impact is high and the communication between train and Central Health/Attack manager is lost. Unique commands are those like the initialization process. It happens once at the beginning of an experiment. The history log behaviour is another elementary command because it is only used by one single agent (CHAM).
Parallel commands are those that are performed from several mostly similar agents that have to do the same thing at the same time. For example, the authentication or keep alive process of all Railway Health/Attack Manager and Train Health/Attack manager. These behaviours are used by a lot of agents and it is urgently necessary that they are performed at the same time respectively in the same simulation phase within one simulation cycle. So, all affected agents have to send a keep alive signal at the beginning of a simulation run. It can strongly bias the simulation result if one or more agent send this signal at the beginning and the remaining agents at the end of a simulation cycle. Since the arrangement of commands has high impact on the outcome, it is useful to keep this in mind for further simulation scenarios.

The sequence command defines that specific behaviours have to be processed in a concrete order. For example, the simulation has to perform behavior A, then B, then C. This is necessary in our simulation because we have to initiate the simulation before we can start the agent behavior execution. So, here we define a concrete order of behavior performance.

5 Modeling SECRET in SyMo and Implementing SECRET in Ready

5.1 Modeling of railway network topology and communication architecture

Here we describe the actual railway network topology [11] in the European railway network (maybe only a link to other deliverables) and show how we inherited a simplified model for SyMo. This model is built in that way that we can run all considered attack scenarios/simulations with this one topology. The railway network topology consists of sectors adjacent to each other. The topology is similar to a honeycomb structure. In a sector one find rail tracks and branches. The communication architecture runs over data transceiver using a specific transmission frequency range. One sector defines the coverage range of its data transceiver that is located in the middle of a sector. Figure 1 shows the railway topology that we use in SyMo and Ready to run the scenario simulations. The track lines define the way a train can use to reach its destination. If a train wants to drive from a sector on the left side to the right side there are only some specific routes foreseen.

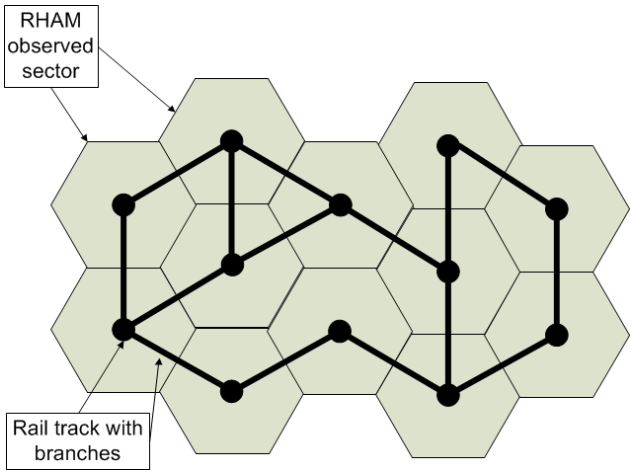


Fig. 8: Railway Topology modeled in SyMo

5.2 Modeling of Dynamic Protection System

Here we describe how the existing railway topology is extended by the Dynamic Protection System concept [12].
The main aim of the Dynamic Protection System (DPS) is to detect attacks that are performed in the protected infrastructure and thus be able to react to overcome them. The detection system consists of the resilient communication architecture which replaces the existing one and observes the communication state. And there is a new resilient health and attack management subsystem which initiates counter measures in case of an electromagnetic attack.
The resilient communication architecture itself contains a detection system based on a sensor network that allows real-time detection of electromagnetic attacks that occur in the railway infrastructure. Furthermore it includes the Multipath Bidirectional Communication System (MBCS) which is responsible for the recovery of an impaired communication between train and central health and attack manager.

The resilient health and attack manager subsystem (HAM) is separated into three similar health and attack manager. They are located at the command center (CHAM), onboard on the train (THAM) and on the railway infrastructure (RHAM).

- The CHAM is the central health and attack manager which controls the underlying THAM and RHAM
- The THAM is connected to the CHAM via wireless communication interfaces
- The RHAM is connected to the CHAM via wired communication interfaces

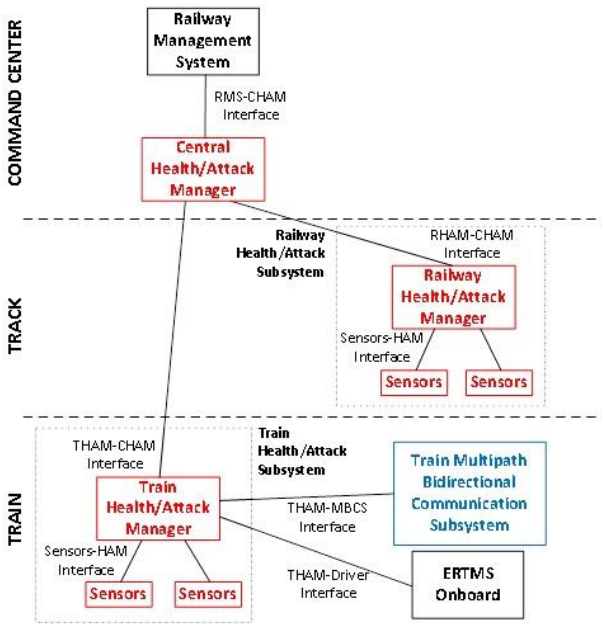


Fig. 9: Architecture of the detection system. From Top to bottom: the command center with its central health/attack manager, the track observing system with its railway health/attack manager, and the train observing system with the train health/attack manager

All health and attack manager systems have similar functionalities but some additional varying specific features that are depending on their specific location. The health and attack manager has to decide appropriate actions during an electromagnetic attack situation. The RHAM and THAM are similar systems with similar functionalities. However, the CHAM is more complex since it governs the overall resilient communication system.
The CHAM is on top of the health and attack manager hierarchy. It uses more functionalities to observe, monitor and control the whole underlying systems.

5.2.1 Components

Here we describe which components of the railway network topology and communication infrastructure we use for the SyMo model and Ready simulation. Why we use them and why we renounce specific components. Some components are urgently necessary and others are not necessary for the simulation.

5.2.2 Main components

Acquisition system sensors

The sensors of the acquisition system provide input information to be processed by the Acquisition System Analyzer. The sensors are arranged in a sensor network. Each health and attack manager monitors one or more sensors. The RHAM controls more sensors than the train. A train has only a limited area. To observe this area only some sensors, maybe at the front and at the end of the train are necessary. On the track the sensor network has to observe a larger area. So, it is urgent necessary to deploy more sensors. The information provided by sensors will vary depending on the kind of sensor and, for example, it may be the radio frequency signal received by an EM sensor or even information like the latency, bit error rate or number of retransmissions. This raw information will be received and processed by the Acquisition System Analyzer to detect

the presence of EM attacks.

Acquisition system analyzer

The Acquisition System Analyzer is mainly responsible for processing the information provided by sensors to detect the presence of EM attacks. It must also control sensors and establish the communication with the Health/Attack Manager (HAM) to inform about changes detected in the state of the EM environment detected. Since the acquisition system analyzer and the system sensors are integrated parts of the health and attack manager it is not required to model this component as an individual agent in SyMo. They are supposed not to be affected by EM attacks because they are using wired and shield communication links between the components.

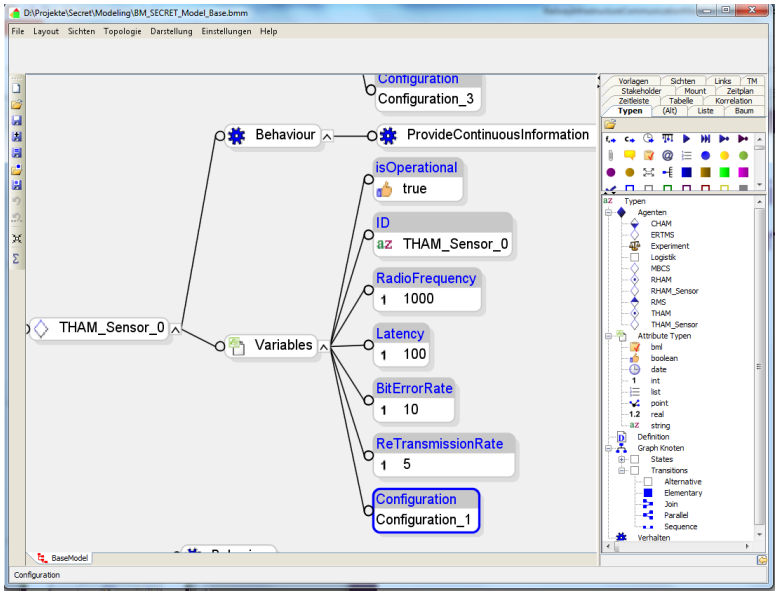


Fig. 10: Sensor model in SyMo

Health and attack manager

The main purpose of the HAM is to collect the electromagnetic attack reported by the acquisition system and process them according to the configuration set by the CHAM.

CHAM

The Central Health/Attack Manager (CHAM) has two well-differentiated functions that must accomplish. Firstly, it must perform the orchestration of detection subsystems to allow the reception of reports of attack and to improve the detection capabilities of isolated detection subsystems. The CHAM creates a comprehensive view on the entire railway network topology and the communication architecture. Secondly, it must provide an interface to allow the management and operation tasks on the detection system.

RHAM

The RHAM is located directly under the CHAM. It is responsible for reporting of attacks and the current electromagnetic state of the track sensor network.

THAM

The THAM is located directly under the CHAM. It is responsible for reporting of attacks and the current electromagnetic state of the onboard sensor network.

Multipath Bidirectional Communication System

The multipath bidirectional communication system (MBCS) is the most important interface in the communication architecture. With this component it is possible to re-establish the communication between train and CHAM in case of an electromagnetic attack. As needed the MBCS has access on different transmission technologies which can be changed on

the fly.

Attack Device

Indeed, this component is not part of the dynamic protection system. The attack device is an absolutely necessary component to test the dynamic protection system. It defines the way how the dynamic protection system will be attacked in a scenario simulation (fig. 11).

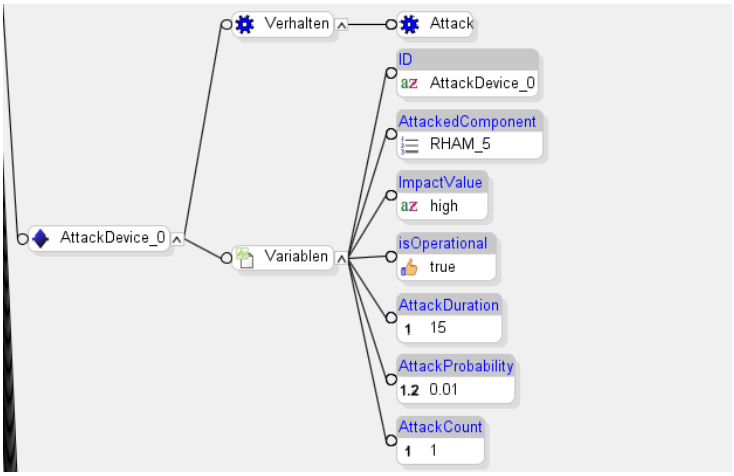


Fig. 11: Attack device model in SyMo

5.2.3 Interfaces

The interfaces describe which component of the communication architecture in the dynamic protection system can exchange data with which other component. These interfaces are not implemented in SyMo. This is not necessary because the interfaces themselves have no concrete behaviour or specific attributes which have to be modelled. They are implicated by the architecture hierarchy and get their function by the JAVA implementation in Ready. For the scenario simulation the interfaces are invisible. Thus it becomes clear that they are a tool for specific tasks of the main components, we will briefly explain what purpose they fulfill.

AS-DS Interface

This interface is the interface between the Acquisition System (AS) and the Detection System (DS), or more precisely, the interface between the Acquisition System Analyzer and the Health/Attack Manager (HAM). The main objective of this interface is to report the EM attacks detected by the AS to the DS. It is required to define the format of an EM attack report message. At least that report message should include information about the EM attack, the location of the attack and the attack severity.

HAM-CHAM Interface

This interface is the interface between the Central Health/Attack Manager (CHAM) and the remotes Health/Attack Managers (HAMs). It is an internal interface of the Detection System (DS) and thus would be specified by the WP4 but it would be conditioned by the EM attack report specification and other requirements of the AS-DS Interface.

HAM-RHAM Interface

This interface is the interface between the Central Health/Attack Manager (CHAM) and the remotes Health/Attack Managers (HAMs). It is an internal interface of the Detection System (DS) and thus would be specified by the WP4 but it would be conditioned by the EM attack report specification and other requirements of the AS-DS Interface.

HAM-THAM Interface

This interface is the interface between the Central Health/Attack Manager (CHAM) and the

remotes Health/Attack Managers (HAMs). It is an internal interface of the Detection System (DS) and thus would be specified by the WP4 but it would be conditioned by the EM attack report specification and other requirements of the AS-DS Interface.

DS-MCS Interface

This interface is the interface between the Detection System (DS) and the Multipath Communication System (MCS), or more precisely, between the Health/Attack Manager (HAM) and the Multipath Communication System (MCS) of a train. It is an external interface of the DS and would be designed and developed in the WP4. The aim of this interface is to establish an association between both systems, so that the detection system can manage the behaviour of the Multipath Communication System according to the detected EM attacks.

CHAM-RMS Interface

This interface is the management interface of the Detection System (DS). It shows that the information captured by the CHAM has to be sent to the railway management staff. In our simulation this information is displayed in our graphical user interface.

5.2.4 Overview about HAM

The health and attack manager has two main functionalities.

- Collect data about an electromagnetic attack reported by the sensor network (furthermore called acquisition system) which observes the electromagnetic state.
- Process the data according to the configuration set by the central health and attack manager.

The health and attack manager must establish and keep a communication session with the CHAM. Stationary fixed components like the railway health and attack manager deployed on the track have a wired communication interface with the CHAM based on the IP protocol. If the health and attack manager is installed on the train there are several alternatives. On the one side one could install the dynamic protection system without the multipath bidirectional communication system. In this case the THAM and driver information system would have separated wireless communication interfaces. On the other side the one could install the dynamic protection system including the multipath bidirectional communication system. In this case there is only one wireless communication interface between MBCS and CHAM because the on-board health and attack manager and the driver information system are both wired connected to the MBCS.

5.2.5 Central Health/Attack Manager CHAM

In the following (fig. 12) you can see the concrete specification of the central health and attack manager that we use in SyMo to simulate the mentioned scenarios at all. In its central position of the communication hierarchy the central health and attack manager is responsible for several crucial tasks. These tasks are interpreted as several behaviours which we have to simulate.

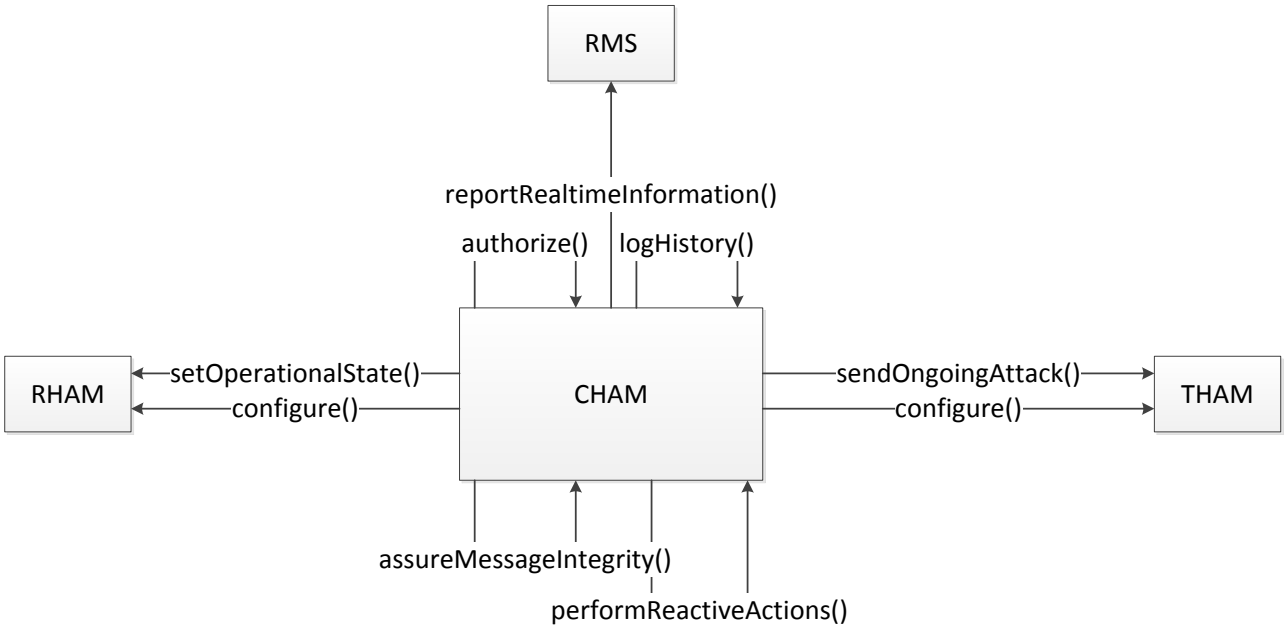


Fig. 12: CHAM agent communication architecture

The CHAM is connected to the railway management system. The railway management system (RMS) is the information system of the railway management staff. It is a graphical user interface where all information captured from the CHAM is shown. With this GUI the personnel can monitor what happens in the communication flow of the trains or in which sector an electromagnetic attack is detected. The GUI displays the railway topology mentioned before. It is updated in real-time every second.
Additionally the CHAM is connected to the THAM. In figure 12 you can see that there are two tasks which define what the CHAM can apply to the train. At the beginning of each simulation the train receives a configuration about what to do in which cases. For example, when an electromagnetic attack occurs the train checks its configuration and initiates a concrete counter measure. In the case that the CHAM is informed about an ongoing attack in a sector which a train may enter soon on its route the CHAM informs the train about the actual situation.
The CHAM is also connected to all railway health and attack manager. In fig. 12 you only see one RHAM but indeed there are more RHAM distributed in the railway topology. There are also two different tasks. Each RHAM gets a configuration similar to the train configuration with information about possible countermeasures. Beyond that the CHAM can change the operational state of a specific RHAM. If an attack or something else is happening and a compromised RHAM transmits continuously ambiguous data to the CHAM its can decide to shut down the affected RHAM.
At least there are some CHAM tasks that are performed internally within the agent. The authorization request of each underlying agent is executed by the CHAM. The CHAM monitors the electromagnetic state of the complete railway topology and communication system. The CHAM knows which RHAM and THAM is currently working or sending data about ongoing electromagnetic attacks. To avoid that a fake health and attack manager registers at the CHAM each component has to be identified by the CHAM. During simulation runtime the information transmitted by the components has to be checked if its content may be compromised during an attack. If someone interferes the original signal and instead sends a wrong signal that could lead to incorrect responses on the part of the CHAM. Whatever the CHAM does is controlled via the “performReactive actions” method. This is the most important behaviour of the agent. Here we decide what to do in which case depending on the input parameters. For example, here the decision is made to

change the communication path interface of the CHAM/THAM communication. Every performed action or event that is executed or captured by the CHAM will be logged in a file for later analysis. Figure 14 shows the SyMo implementation of the CHAM behaviours.

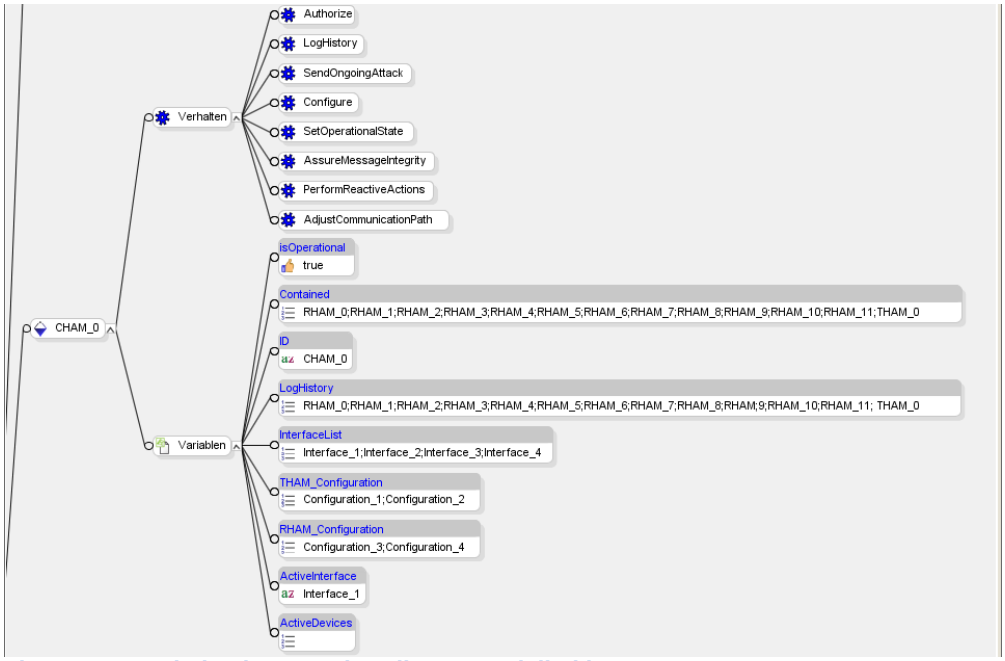


Fig. 13: CHAM behaviours and attributes modelled in SyMo

The following list shows a detailed description of the modelled behaviours

CHAM Agent Behaviour

- reportRealtimeInformation()
  - Inform the railway management system about electromagnetic state changes in the detection network (sensor failures, sensors switched on/off, EM attack information, communication lack)
- configure()
  - Set maximum update interval for registration for specific RHAM/THAM
  - Set options of the electromagnetic attacks to be detected (thresholds, patterns) and give some countermeasures about what to do when an attack occurs
- setOperationalState()
  - Set the RHAM/THAM into operational state after successful identity verification
- assureMessageIntegrity()
  - Verification of a message
- authorize()
  - Verification of the identity of RHAM & THAM during registration process
- performReactiveActions()
  - Assess RHAM/THAM information and decide which action has to be performed The CHAM has to decide which countermeasures should be initiated edepending on the attack severity and train location
- sendOngoingAttack()
  - Inform THAM about ongoing attack so the MBCS can perform convenient actions
  - Broadcast attack information to specific railway topology agents next to the

attacked sector and to affected trains
- logHistory()
  - Log registration of a railway & train health/attack subsystem
  - Log state changes in the communication network
  - Log keep alive re-acknowledgement
  - Log attack information
- adjustCommunicationPath()
  - At the beginning of each simulation run the connection between train and CHAM is established. In case of an attack the THAM switches communication to any of the other interfaces.

The following list shows a detailed description of the modelled attributes.

CHAM Agent Attributes
- isOperational
  - Sets the state of this agent to operational/not operational
- contained
  - Holds a list of known agents to verify agents
- ID
  - Identification number
- InterfaceList
  - List of available redundant interfaces. In case of an EM attack the active interface switches to one of these interfaces depending on the MBCS state and the attacked interface.
- THAM_Configuration
  - Holds a list of countermeasure configurations defined for THAM about what to do in case of an EM attack
- RHAM_Configuration
  - Holds a list of countermeasure configurations defined for RHAM about what to do in case of an EM attack
- ActiveInterface
  - Contains the currently used communication interface in the multipath bidirectional communication system
- ActiveDevices
  - Contains all active and not attacked RHAMs/THAMs to represent a complete view of the railway topology
- LogHistory
  - Saves all Data captured in a simulation run including the timestamp.

5.2.6 Train Health/Attack Manager THAM

Here we describe which concept is behind the train health and attack manager specification. The train health and attack manager has several main functionalities.
- Observe the electromagnetic state of the train using its sensor network to detect attacks
- Report attack information to CHAM
- Inform driver about ongoing attack
- Switch communication interface between train and CHAM

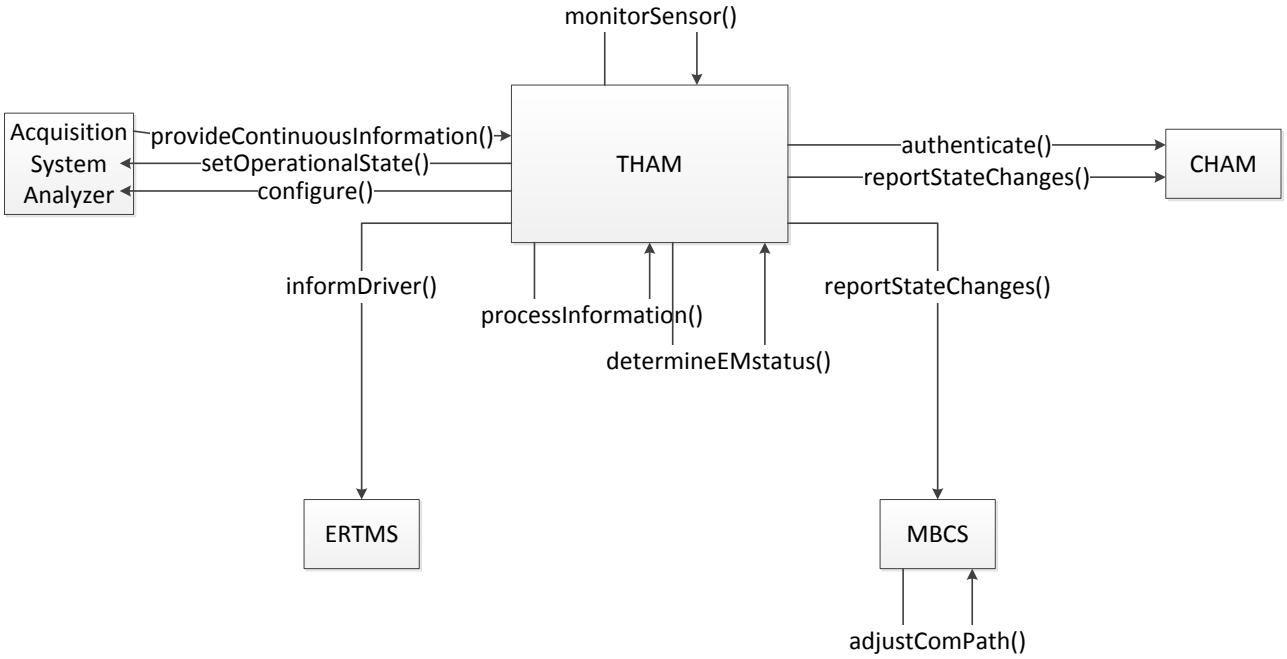


Fig. 14:THAM agent communication architecture

These objectives are achieved by the use of different agent behaviours of the train health and attack management system. As you can see in figure 8 the THAM is connected to four other components of the dynamic protection system. The sensor network link is similar to the CHAM description mentioned above. With the CHAM the train communicates about the authorization request and state change report. Each train health and attack manager has to register or authenticate at the CHAM. For example, the simulation starts and the CHAM does not know which other components are present. So the THAM registers as a new component. The other behaviour reports the electromagnetic state of the train in the case of an attack. If no attack occurs for a longer time, the CHAM could interpret this as a communication loss. To avoid this misinterpretation the THAM is forced to periodically send a keep alive signal.

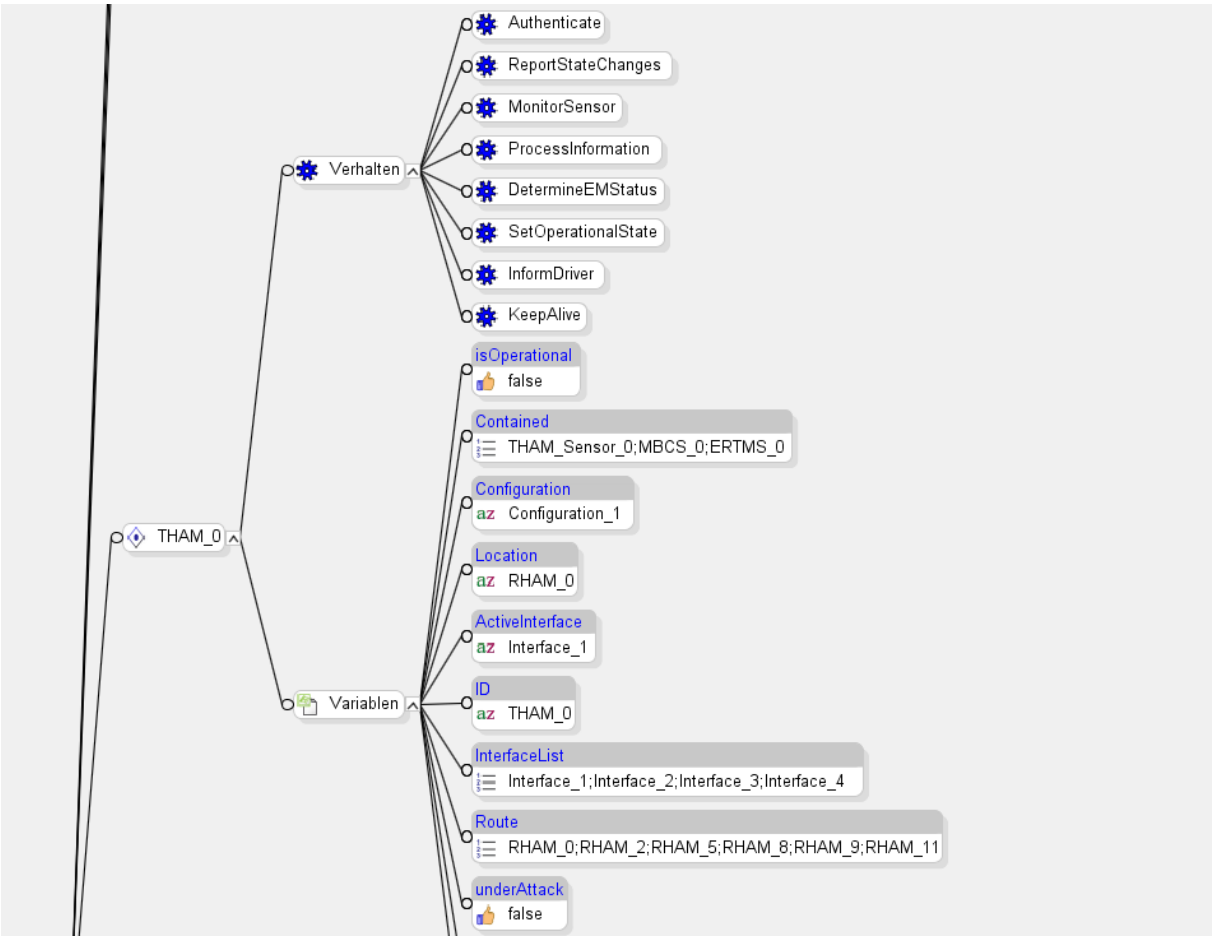


Fig. 15: THAM behaviours and attributes modelled in SyMo

The following list shows a detailed description of the modelled behaviours.

THAM Agent Behaviour

- authenticate()
  - Send unique identification ID to CHAM to initiate registration process
- keepAlive()
  - Sends a periodical signal to the CHAM to re-acknowledge that it is still working and not attacked
- monitorSensor()
  - Continuously observe sensor input interface for new information
- processInformation()
  - Decide between attack/failure information
- determineEMstatus()
  - Assess sensor input based on configured options for EM attack detection
- reportStateChanges()
  - In case of changes within the THAM immediatly send the new state to CHAM (sensor failure, detected EM attack, communication lack)
  - Additionally report state changes to the MBCS
- setOperationalState()
  - In case of sensor failure turn sensor off
- informDriver()
  - Send attack message to train driver
- Sensor Agent Behaviour

- provideContinuousInformation()
  - Capture sensor data based on the configured options
- MBCS Agent Behaviour
- adjustComPath()
  - In case of EM attack or communication lack we change the communication path according to the EM attack and the current configuration provided by the CHAM

The following list shows a detailed description of the modelled attributes.

THAM Agent Attributes

- isOperational
  - Sets the state of this agent to operational/not operational
- contained
  - Contains a list of all connected acquisition system analyzers, additionally the multipath bidirectional communication system and ERTMS interface
- configuration
  - Holds a list of configuration information provided by CHAM. This list is filled after a successful registration process
- location
  - Shows the actual train location within the railway topology by using the RHAM section
- ID
  - Identification number
- ActiveInterface
  - Contains the currently active communication interface in the multipath bidirectional communication system
- underAttack
  - Shows the actual attack state of this agent. At the beginning it is not under attack
- KeepAliveTimer
  - Contains a value (e.g. 5 seconds) about the periodically processed keep-alive timer for this agent. For running different scenarios it could be necessary to change this value
- InterfaceList
  - List of available redundant interfaces. In case of an EM attack the active interface switches to one of this interfaces depending on the MBCS state and the attacked interface. There are some process rules implemented to guarantee that CHAM and THAM automatically switch to the identical communication interface (e.g. both switch from GPRS to LTE)
- Route
  - Shows the route that the train has to drive during a simulation run
- isDriving
  - Sets the state of a train to driving/not driving
- **Sensor Agent Behaviour**
- provideContinuousInformation()
  - Capture sensor data based on the configured options
- **MBCS Agent Behaviour**
- adjustComPath()

- In case of EM attack or communication lack we change the communication path based on the commands provided by the THAM

5.2.7 Railway Health/Attack Manager RHAM

The railway health/attack manager (see [12]) is responsible for observing the track, evaluating the sensor data and reporting attack events to the CHAM via a *wired* communication connection. Due to the wired communication there is no need for RHAMs for a multi-channel communication. Figure 17 shows the communication paths between sensors and the CHAM and which behaviours are relevant for the simulation experiment.

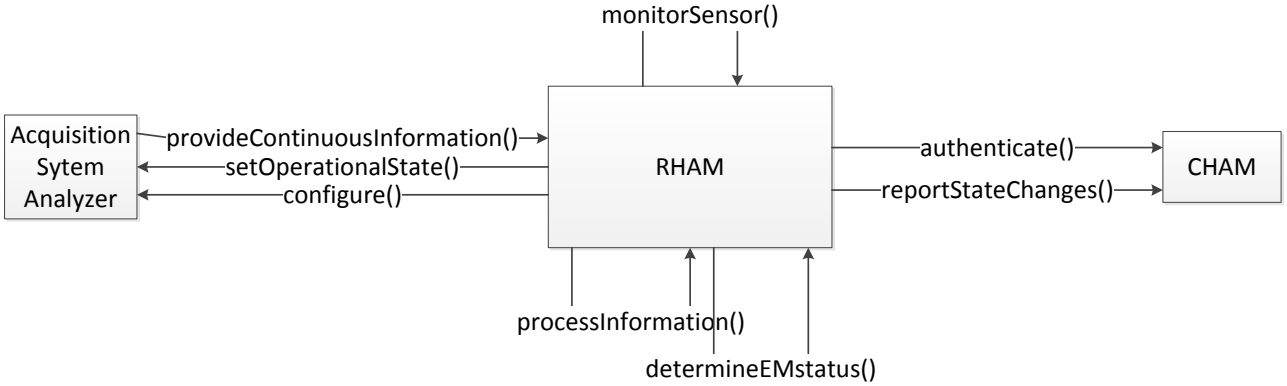


Fig. 16: RHAM agent communication architecture

Figure 18 shows the SyMo Model of one RHAM with the mentioned behaviours from the previous description in fig. 17. Additionally you see in SyMo the parameters which are necessary for the simulation run and the performance of all behaviours.

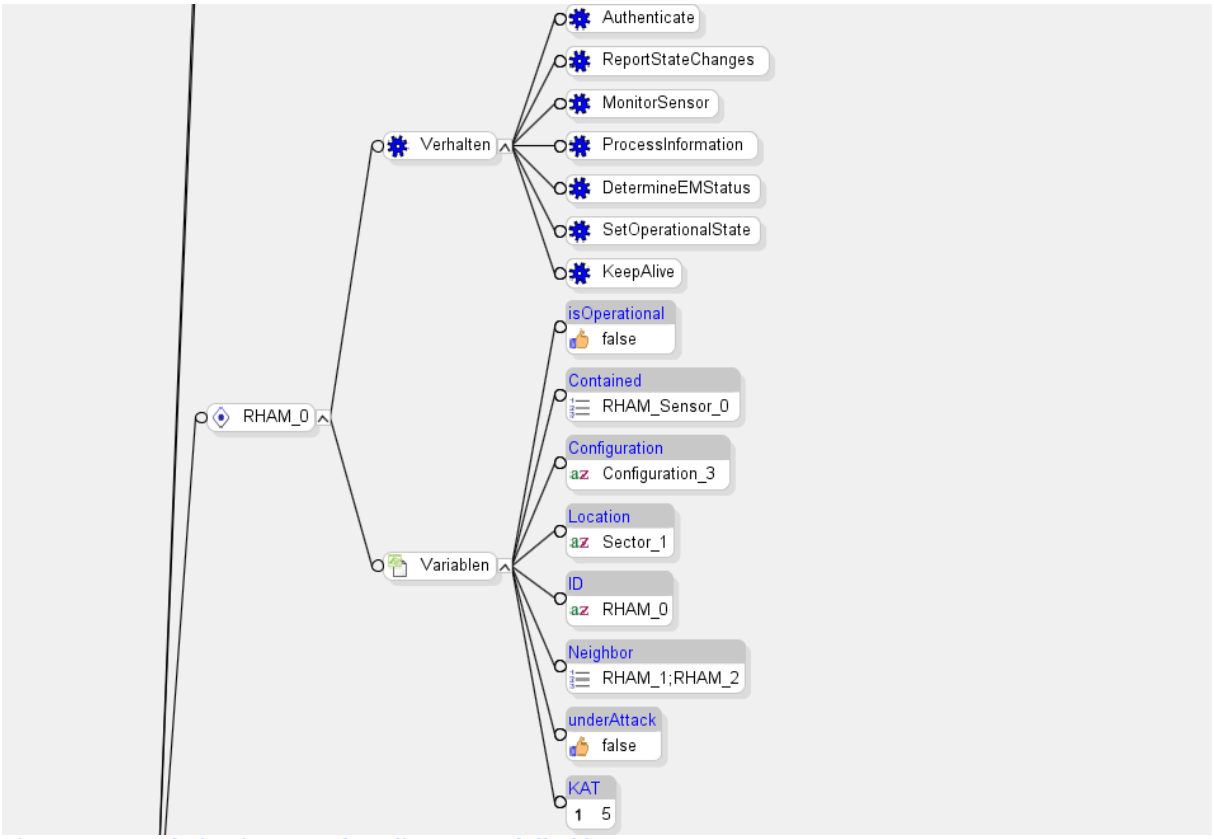


Fig. 17: RHAM behaviours and attributes modelled in SyMo

The following list shows a detailed description of the modelled behaviours

RHAM Agent Behaviour

- authenticate()
  - Send unique identification ID to CHAM to initiate registration process
- monitorSensor()
  - Continuously observe acquisition system analyzer input interface for new information
- processInformation()
  - Decide between attack/failure information
- determineEMstatus()
  - Assess sensor input based on configured options for EM attack detection
- reportStateChanges()
  - In case of changes within the RHAS immediately send the new state to CHAM (sensor failure, detected EM attack, communication lack)
- setOperationalState()
  - In case of sensor failure turn sensor off
- keepAlive()
  - Sends a periodical signal to the CHAM to re-acknowledge that it is still working and not attacked

Acquisition System Analyzer Agent Behaviour

- provideContinuousInformation()
  - Capture sensor data based on the configured options

The following list shows a detailed description of the modelled attributes.

RHAM Agent Attributes

- isOperational
  - Sets the state of this agent to operational/not operational
- contained
  - Holds a list of all connected sensors
- configuration
  - Holds a list of configuration information provided by CHAM. This list is filled after a successfull registration process
- location
  - Shows the agents‘ location within the railway topology
- ID
  - Identification number
- Neighbor
  - Holds a list of the directly connected neighbor health/attack manager. By interpreting this attribute in each RHAM the CHAM can create a complete view over the railway topology.
- underAttack
  - Shows the actual attack state of this agent. At the beginning of a simulation it is not under attack. You can say that the scenario is in a secure state.
- KeepAliveTimer
  - Contains a value (e.g. 5 seconds) about the periodically processed keep-alive timer for this agent. For running different scenarios it could be necessary to change this value.

5.2.8 Attack Device

Here we define which concept is behind the attack device specification. The purpose of the device is to test the resiliency of the dynamic protection system against electromagnetic attacks. The railway topology model and the new communication architecture model are the base for this. With that agent we now want to show how the complete system behaves in case of an attack.

Attack Agent Behaviour

There is only one behaviour: Attack. This behaviour is triggered during simulation runtime by the attribute: AttackProbability.

- attack()
  - This behaviour processes the attack moment

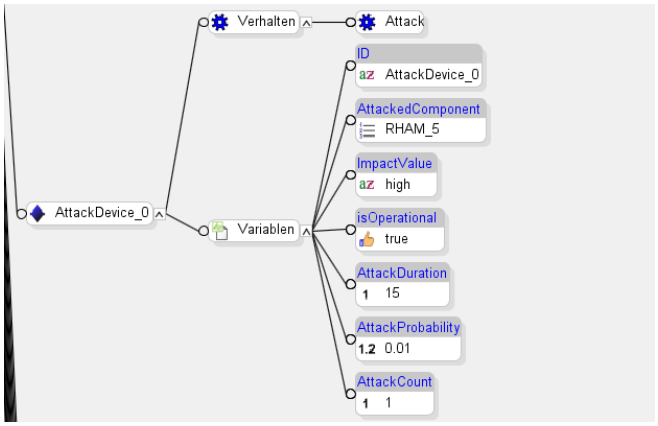


Fig. 18: Attack device in SyMo

Attack Agent Attributes

- ID
  - Identification number
- AttackedComponent
  - Contains the components which we want to attack in a simulation. This parameter is a list because we want to test different attack scenarios where more than one component can be affected by an electromagnetic attack.
- ImpactValue
  - The impact value defines the severity of an electromagnetic attack. We distinguish low, medium and high impacts. Low impacts are defined as weak frequency interferences which are detected and interpreted as a possible attack but don't result in a communication loss. Medium impacts are defined as a stronger interference which can result in the loss of data packages or result in increasing transmission latency. This case is not dangerous at the moment can may result in a communication loss. The high impact is a very strong interference which immediately interrupts the communication.
- isOperational
  - Sets the state of this agent to operational/not operational
- AttackDuration
  - This value sets the time frame about how long an attack will take. The value represents the time in seconds.
- AttackProbability
  - Send unique identification ID to CHAM to initiate registration process
- AttackCount
  - Send unique identification ID to CHAM to initiate registration process

Which component is attacked can be defined in the model

5.2.9 Multipath Bidirectional Communication System MBCS

The Multipath Bidirectional Communication System (MBCS) (fig. 20) is responsible for reestablishing the communication between the train and CHAM. The MBCS controls several communication interfaces which use different frequency spectrums to transmit and receive data. In case of an attack at least one of these frequencies can be disturbed by the jamming device. This causes a communication loss on train side. The MBCS has to appropriately react on the ongoing attack and has to make sure that the communication to CHAM is reestablished by using another frequency, respectively another interface. In order to guarantee this the MBCS has internal rules for switching between communication interfaces – for instance from interface 1 to interface 2, or from interface 2 to interface 3.

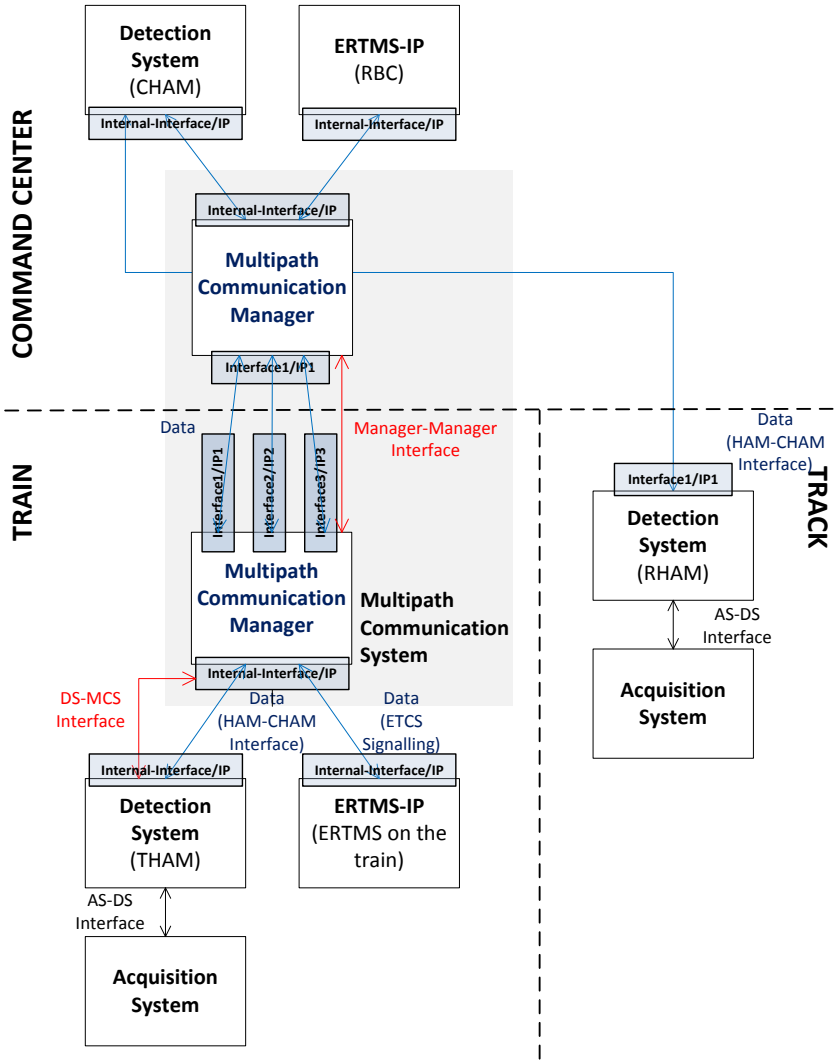


Fig. 19 Architecture of the Multipath Bidirectional Communication System.

5.2.10 Interfaces

Here we describe which interfaces of the communication infrastructure we use in which way for the SyMo model and Ready simulation. The interfaces are part of the new Dynamic Protection System. They control the dataflow between HAM structures and the new MBCS.

6 Adaptation of Use Cases to simulation experiments

6.1 Introduction

The following chapter provides simulation results in different electromagnetic attack scenarios [13]. The base of all simulation runs is the already shown railway topology. (fig. 20) This base topology is enhanced by new components of the Dynamic Protection System. As you can see there are now a lot of railway health/attack manager (RHAM) deployed at the existing railway topology. Each RHAM observes a specific section of the railway topology. Some RHAMs are neighbors of others and some are a little bit distant to others. They are connected via rails where the train can drive (fig.21). This topology is a simplified analogy to the existing railway network. It shows the logical geometry of rails in the real world in a smaller form which is suitable for a – not too complex simulation – but still large enough to obtain meaningful results. We implemented 12 railway health/attack managers arranged in a topology structure to create a gapless covered simulation environment.

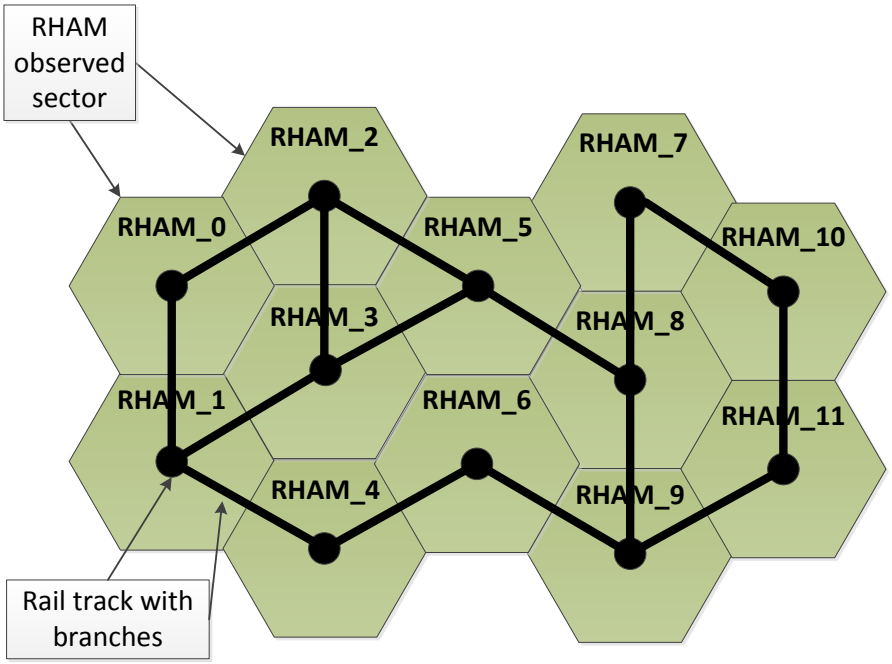


Fig. 20 Simulated railway topology with concrete RHAM allocation. Black points describe the RHAM position and the hexagons surrounding the black points define the observed RHAM area. Black lines specify the route where a train can drive

Here (fig. 21) you can see the modeled driving route of the train. This route is fix modeled in our System Modeler but can be easily modified for other simulation issues. In our simulation experiments the train starts at RHAM_0 section and passes RHAM_2, RHAM_5, RHAM_8, RHAM_9 sections until the train finally reaches its destination at RHAM_11 section.

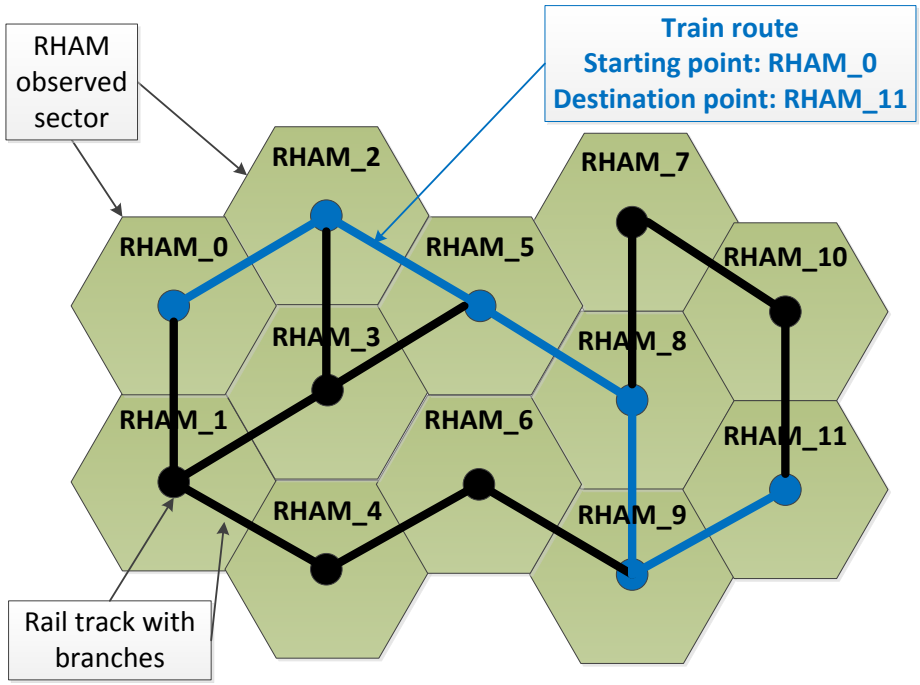


Fig. 21 Simulated railway topology including the simulated train route. Blue lines show the route where the train drives along during a simulation run. Each route has a start point and destination point

For a later specification of different attack scenarios it is also important to show the train (fig.22) within the scenario because in some simulation runs the train will be directly attacked by an electromagnetic jamming device. In some other simulations the train is not directly attacked.

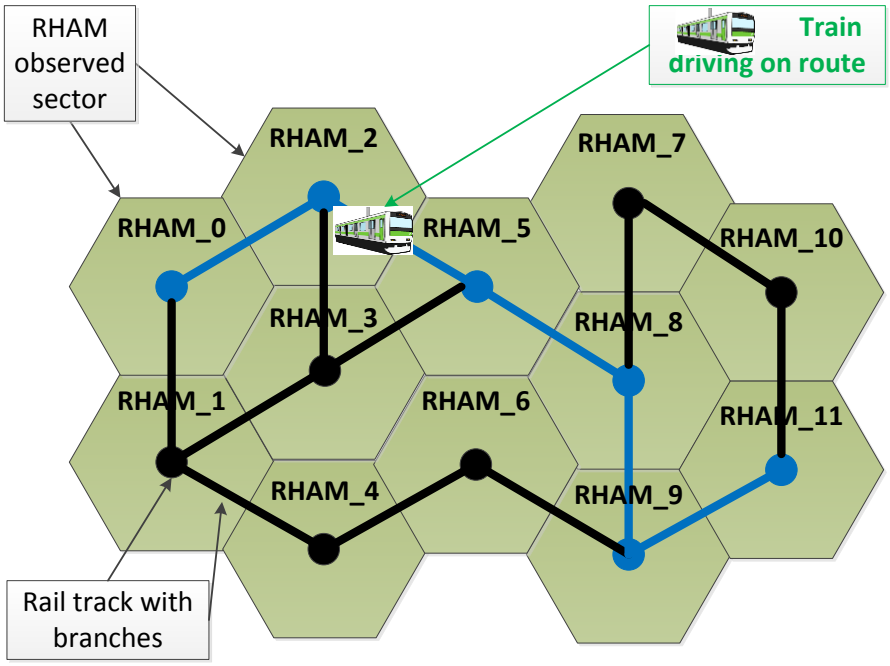


Fig. 22 Integration of the train into the simulated topology. The small train symbol shows that one train is driving along the blue line route

In each simulation scenario we capture time stamps when which behaviour has been executed. The chronological sequence of the behaviours can vary from scenario to

scenario. This results by trigger events which do different things in a different order depending on their triggering input parameter. E.g. in one case the THAM switches its MBCS interface before the CHAM does it and in another case it is reverse because the information chain behaves different based on the simulation environment, attack destination and the trigger events. To represent this we decided to show 2 diagrams of each scenario with a specific view on the complete behaviour process (see e.g. fig. 24 and 25). In diagrams like fig. 25 you see only those behaviours which are executed within the concrete simulation run. So you can perceive the detailed behaviour sequence with information which behaviour follows on another behaviour in temporal relation to each other. This view enables to detect where time delays occur and to evaluate them.
The diagrams like fig. 26 show the same information like figure 25 in another view and additionally the relation to all other simulation experiments. Therefore, there are also information about behaviours and components which are not considered in a concrete scenario. E.g. you can see the RHAM_5 EM attack detection term in this figure although RHAM_5 is not relevant in this model. In other models RHAM_5 is involved into an attack and by comparing these diagrams you can see differences in the behaviour executing sequence and analyze this to evaluate the reasons and results of these differences.

All simulation scenarios are named as a specific SECRET Base Model_X where the X is representative for a continuing number. One Base Model is a concrete simulation model which we have modeled in System Modeler. The difference in the models is that we consider various combinations of input parameters. We set

- the attack location (e.g. the train or one or more railway health/attack manager) to simulate the reaction time when different parts of the Dynamic Protection System are involved
- the attack severity (low, medium, high) to simulate how the strong the attack impact affects the event sequence of counter measures
- the attack duration (10 seconds, 5 seconds, 2 seconds) to simulate the behaviour of the Dynamic Protection System when the attack takes very long or only very short time. In our simulation we have limited the attack time to max 10 seconds as along attack. A real attack can take much longer but this is not necessary to simulate. It would only increase the duration for one simulation run and not provide significant new results.
- the keep alive timer of the train health/attack manager (5 seconds) to simulate which influence this parameter has on the complete reaction time)
- the simulation cycle duration (100 ms)
- the train route (see above) to simulate the train behaviour when it drives along the route and something happens anywhere(onboard or on trackside)

6.2 Scenarios

6.2.1 SECRET Base Model_1

- Attacked device: THAM_0 (train)
- Attack severity: high Impact
- Attack duration: 10 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

- Hint: all values are average reaction time of several simulation runs

SECRET Base Model_1 simulates a scenario where the train is directly attack by a jamming device. As you know, this means that the attack device is onboard the train. At the beginning the attack device is disabled. The train drives along the route and initiated by a random time trigger event the attack device gets enabled and disturbs the communication between the train and the central health/attack manager with a high severity. The attack device is active for 10 seconds and then it is disabled.

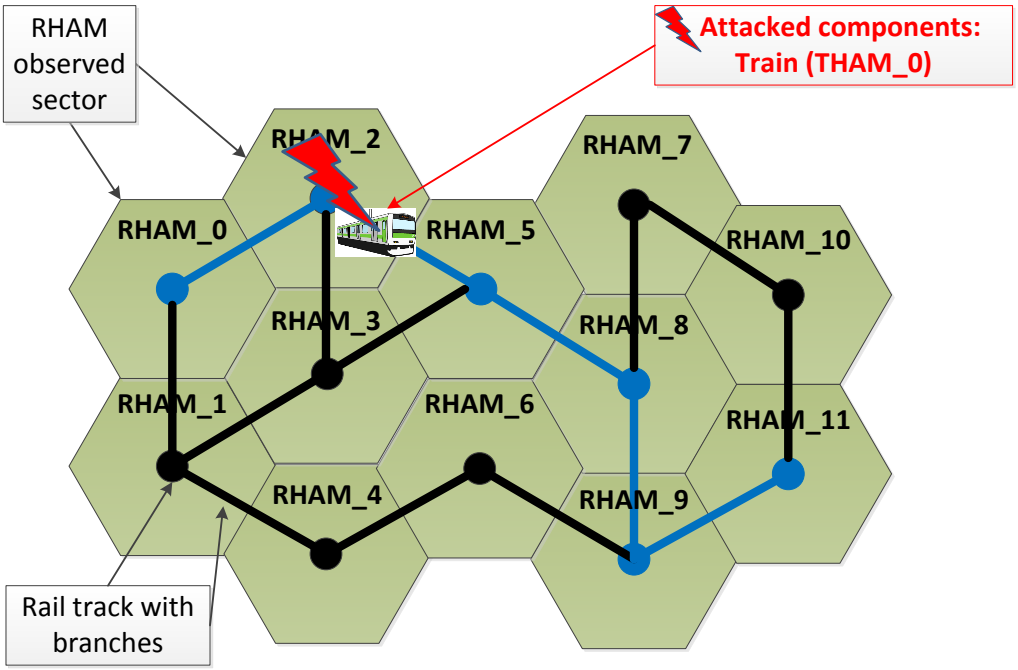


Fig. 23 Simulated Attack Scenario with train under attack. The bolt shows the attack aim in a concrete simulation scenario. This figure describes an attack scenario where the train is directly affected

Result description

- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.061 seconds
- The communication connection between THAM and CHAM is lost 0.089 seconds after the attack event. The connection loss has no impact on the reaction time of CHAM because the KAT is already decreasing. In this case, only the KAT expiration triggers a CHAM reaction.
- It takes 0.223 seconds for the THAM to detect the EM attack. The THAM reacts directly on the connection loss. This results in an
  - *detection time = Com link loss (0.089s)+ Attack detection THAM (0.133s) = 0.223s*
- Because of the connection loss the CHAM cannot be informed by the THAM about the ongoing attack. So, the CHAM detects the EM attack after:
  - *KAT expiration (5.061s) + Detection time (0.127s) = 5.188s*
- The THAM activates its MBCS 0.352s after the attack has been detected by calculating
  - *Com link loss + Attack detection Time + THAM/MBCS activation time*

- The CHAM activates its MBCS 0.159 seconds after the attack has been detected by calculating
  - *KAT + Attack detection time + MBCS activation time = 5.346s*
- In parallel to that the CHAM informs other HAMs 0.164s after the attack has been detected by calculating
  - *KAT + Attack detection time + HAM information time = 5.351s*
- The reestablishment of communication is done after further 0.156s by calculating the specific corresponding times which result in the highest overall reaction time. Here the CHAM is very late with its attack detection moment. Communication is reestablished after
  - *MBCS activation time + Com re-establishment time = 5.502s*
- Conclusion: When the train is attacked and the communication is lost then it takes 5.502s until the communication between train and CHAM is reestablished. Although the train early detects the attack and reacts in a proper way to immediately switch the MBCS interface there is a long time delay caused by 5-seconds-keep-alive-timer within the CHAM. Because there is no opportunity for the CHAM to detect the attack as soon as the train the CHAM has to wait for KAT expiration before it can initiate the MBCS activation on CHAM side. This behaviour recurs in each scenario simulation. If we leave the KAT out of the measurement one can see that the reaction time of each step in the countermeasure process is in a scope of a few tenth-milliseconds.

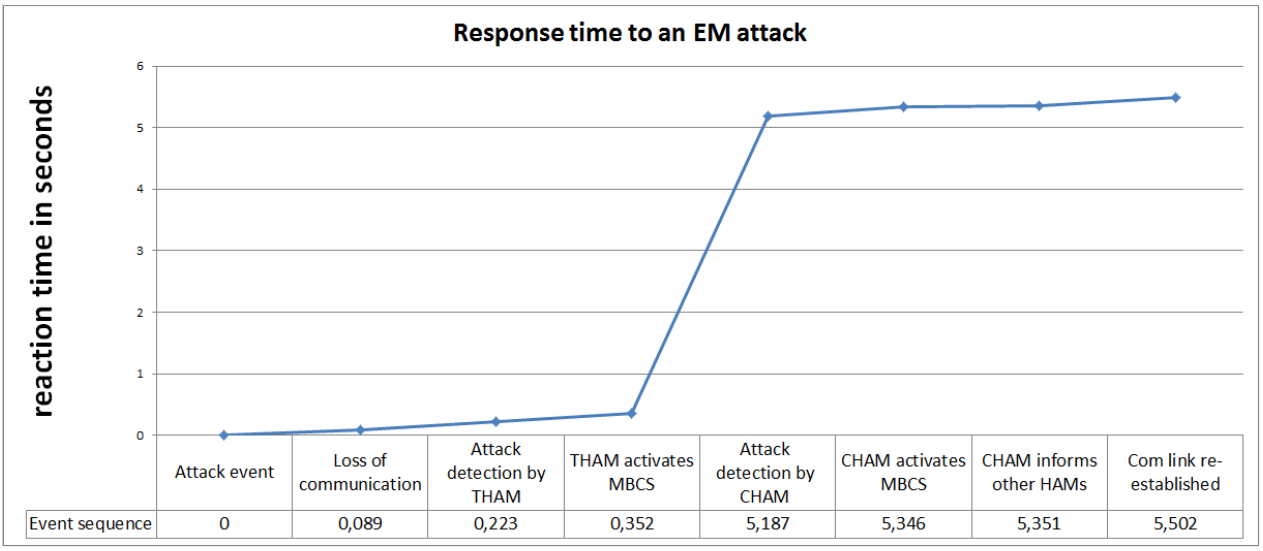
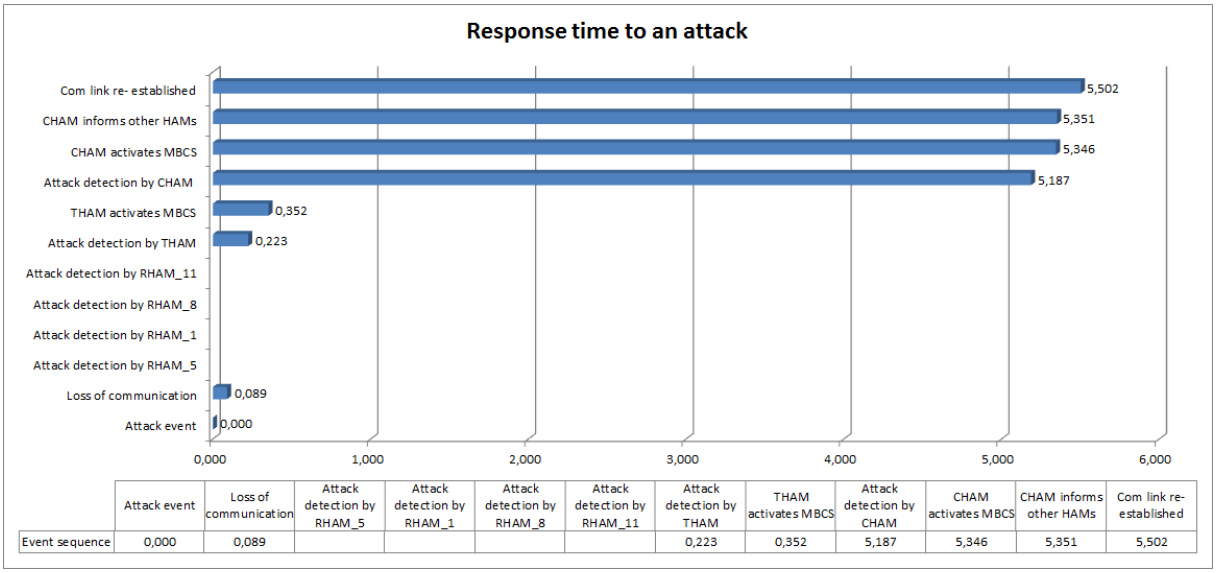


Fig. 24 Response time of the DPS in Base Model 1 (Train under attack) as line with data points. The keep alive timer of the train expires and this results in a long delay until the CHAM detects the attack on its side.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,089					0,223	0,352	5,187	5,346	5,351	5,502

Fig. 25 Response time of the DPS in Base Model 1 (Train under attack) as bar graph

6.2.2SECRET Base Model_10

- Attacked device: THAM_0 (train)
- Attack severity: high Impact
- Attack duration: 5 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_10 describes the same start scenario as in Model_1 with the difference that the attack duration is only 5 seconds.

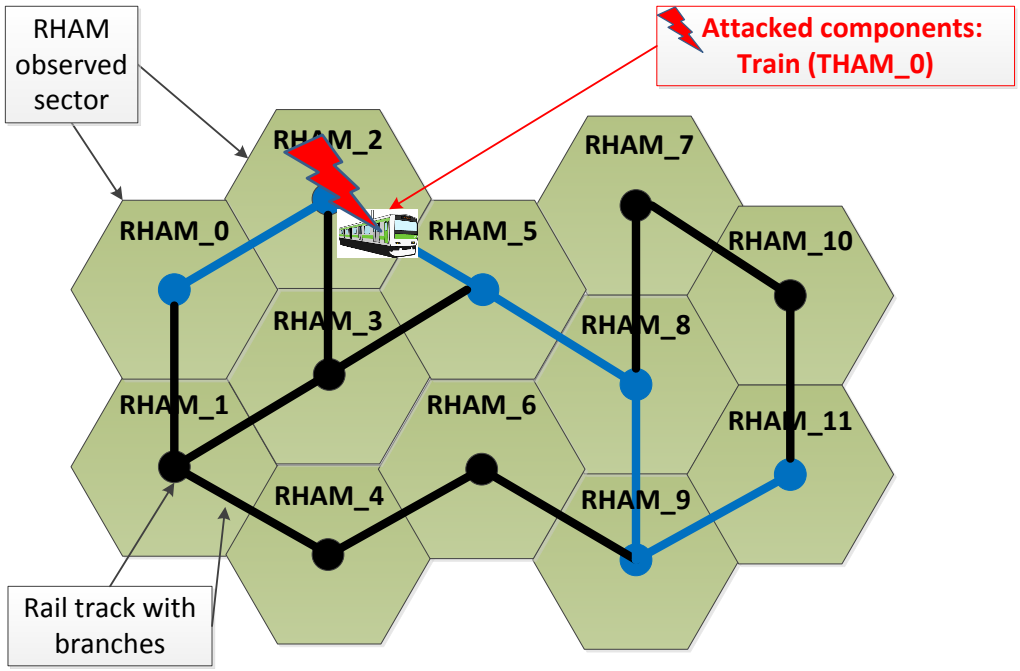


Fig. 26 Simulated Attack Scenario with train under attack

Result description

- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.069 seconds

- The communication connection between THAM and CHAM is lost 0.076 seconds after the attack event. The connection loss has no impact on the reaction time of CHAM because the KAT is already decreasing. In this case, only the KAT expiration triggers a CHAM reaction.
- It takes 0.206 seconds for the THAM to detect the EM attack. The THAM reacts directly on the connection loss. This results in an
    - *detection time = Com link loss (0.076s)+ Attack detection THAM (0.130s) = 0.206s*
- Because of the connection loss the CHAM cannot be informed by the THAM about the ongoing attack. So, the CHAM detects the EM attack after:
    - *KAT expiration (5.069s) + Detection time (0.151s) = 5.221s*
- The THAM activates its MBCS 0.157s after the attack has been detected by calculating
    - *Com link loss + Attack detection Time + THAM/MBCS activation time*
- The CHAM activates its MBCS 0.159 seconds after the attack has been detected by calculating
    - *KAT + Attack detection time + MBCS activation time = 5.372s*
- In parallel to that the CHAM informs other HAMs 0.164s after the attack has been detected by calculating
    - *KAT + Attack detection time + HAM information time = 5.660s*
- The reestablishment of communication is done after further 0.140s by calculating the specific corresponding times which result in the highest overall reaction time. Here the CHAM is very late with its attack detection moment. Communication is reestablished after
    - *MBCS activation time + Com re-establishment time = 5.862s*
- Conclusion: When the train is attacked and the communication is lost then it takes 5.862s until the communication between train and CHAM is reestablished. Although the train early detects the attack and reacts in a proper way to immediately switch the MBCS interface there is a long time delay caused by 5-seconds-keep-alive-timer within the CHAM. Because there is no opportunity for the CHAM to detect the attack as soon as the train the CHAM has to wait for KAT expiration before it can initiate the MBCS activation on CHAM side. This behaviour recurs in each scenario simulation. If we leave the KAT out of the measurement one can see that the reaction time of each step in the countermeasure process is in a scope of a few tenth-milliseconds.

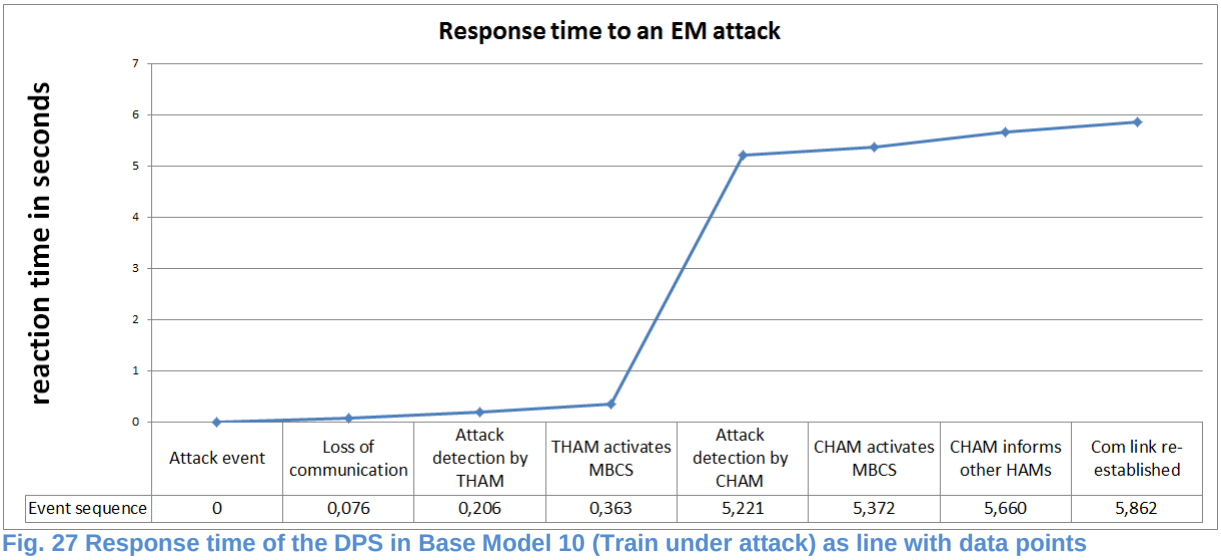


Fig. 27 Response time of the DPS in Base Model 10 (Train under attack) as line with data points

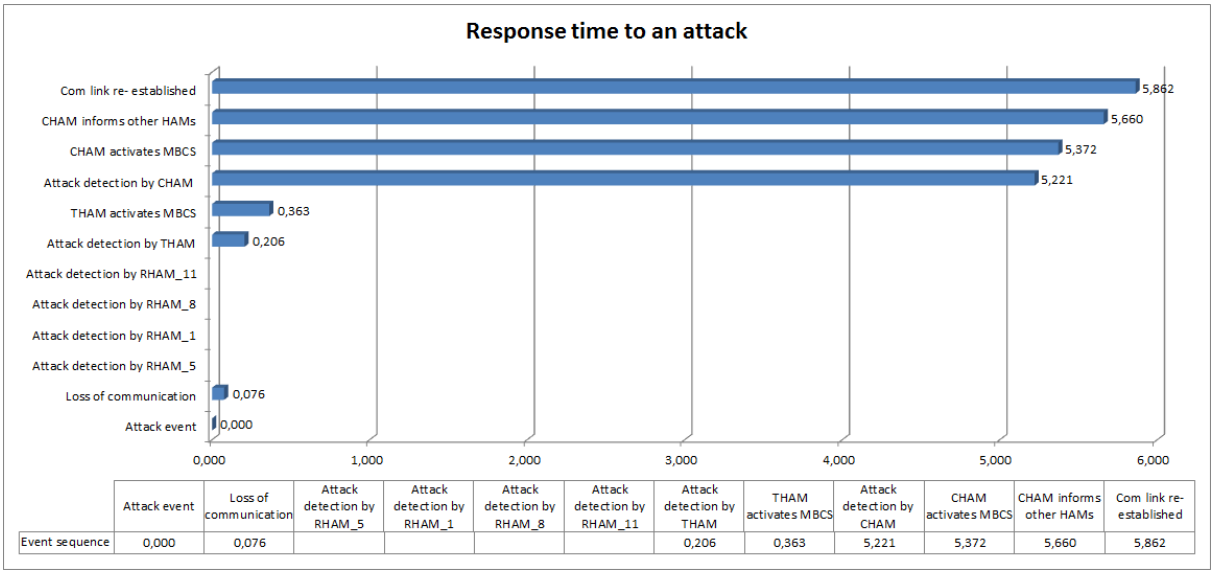


Fig. 28 Response time of the DPS in Base Model 10 (Train under attack) as bar diagram

6.2.3SECRET Base Model_5

- Attacked device: THAM_0 (train health/attack manager))
- Attack severity: high Impact
- Attack duration: 2 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_5 has the nearly same start configuration as Model_1 and Model_10 but here the attack duration is again decreased and set to only 2 seconds.

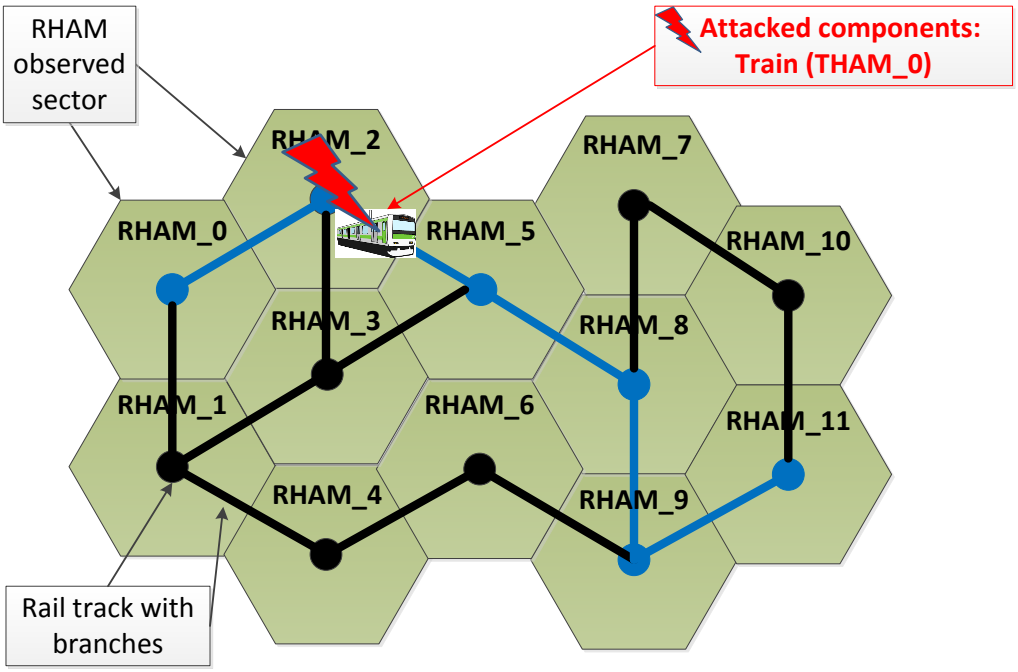
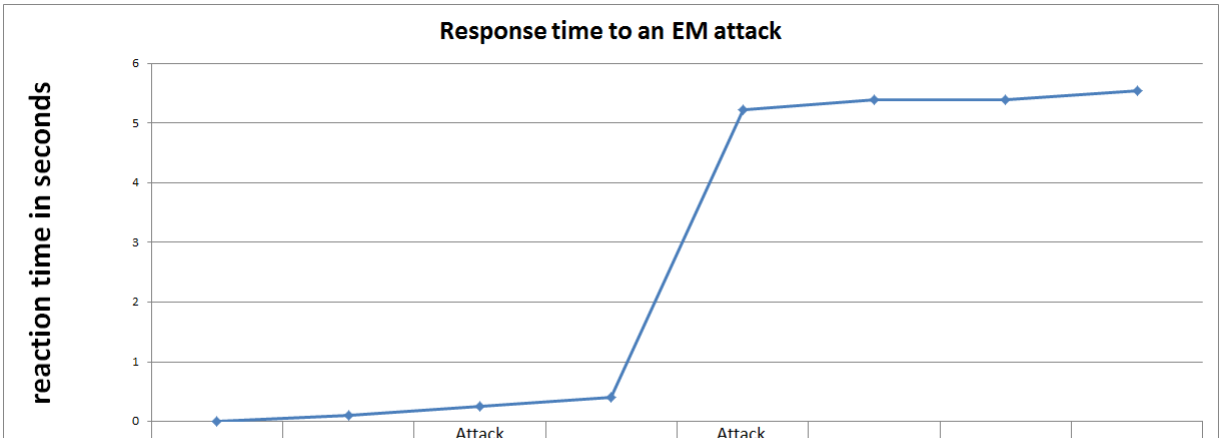


Fig. 29 Simulated Attack Scenario with train under attack

Result description

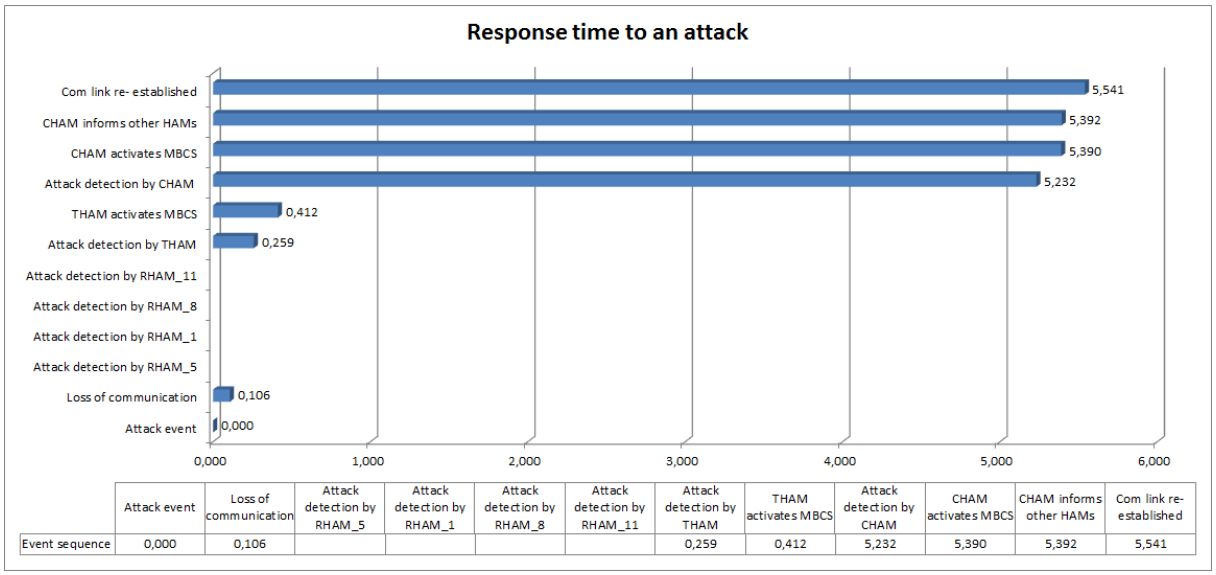
- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.085 seconds
- The communication connection between THAM and CHAM is lost 0.106 seconds after the attack event. The connection loss has no impact on the reaction time of CHAM because the KAT is already decreasing. In this case, only the KAT expiration triggers a CHAM reaction.
- It takes 0.259 seconds for the THAM to detect the EM attack. The THAM reacts directly on the connection loss. This results in an
  - *detection time = Com link loss (0.106s)+ Attack detection THAM (0.153s) = 0.259s*
- Because of the connection loss the CHAM cannot be informed by the THAM about the ongoing attack. So, the CHAM detects the EM attack after:
  - *KAT expiration (5.085s) + Detection time (0.147s) = 5.232s*
- The THAM activates its MBCS 0.153s after the attack has been detected by calculating
  - *Com link loss + Attack detection Time + THAM/MBCS activation time*

- The CHAM activates its MBCS 0.158 seconds after the attack has been detected by calculating
  - *KAT + Attack detection time + MBCS activation time = 5.390s*
- In parallel to that the CHAM informs other HAMs 0.164s after the attack has been detected by calculating
  - *KAT + Attack detection time + HAM information time = 5.392s*
- The reestablishment of communication is done after further 0.140s by calculating the specific corresponding times which result in the highest overall reaction time. Here the CHAM is very late with its attack detection moment. Communication is reestablished after
  - *MBCS activation time + Com re-establishment time = 5.541s*
- Conclusion: When the train is attacked and the communication is lost then it takes 5.541s until the communication between train and CHAM is reestablished. Although the train early detects the attack and reacts in a proper way to immediately switch the MBCS interface there is a long time delay caused by 5-seconds-keep-alive-timer within the CHAM. Because there is no opportunity for the CHAM to detect the attack as soon as the train the CHAM has to wait for KAT expiration before it can initiate the MBCS activation on CHAM side. This behaviour recurs in each scenario simulation. If we leave the KAT out of the measurement one can see that the reaction time of each step in the countermeasure process is in a scope of a few tenth-milliseconds. (An EM attack may stop after, say, 2 sec. In this case the MBCS can reestablish the previous interface.)



	Attack event	Loss of communication	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0	0,106	0,259	0,412	5,232	5,390	5,392	5,541

Fig. 30 Response time of the DPS in Base Model 5 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,106					0,259	0,412	5,232	5,390	5,392	5,541

Fig. 31 Response time of the DPS in Base Model 5 (Train under attack) as bar graph

6.2.4SECRET Base Model_8

- Attacked device: THAM_0 (track health/attack manager)
- Attack severity: low Impact
- Attack duration: 5 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_8 describes a similar scenario in comparison with the previous models. The concrete differences are another attack severity with a low impact on the communication infrastructure and five-second attack duration.

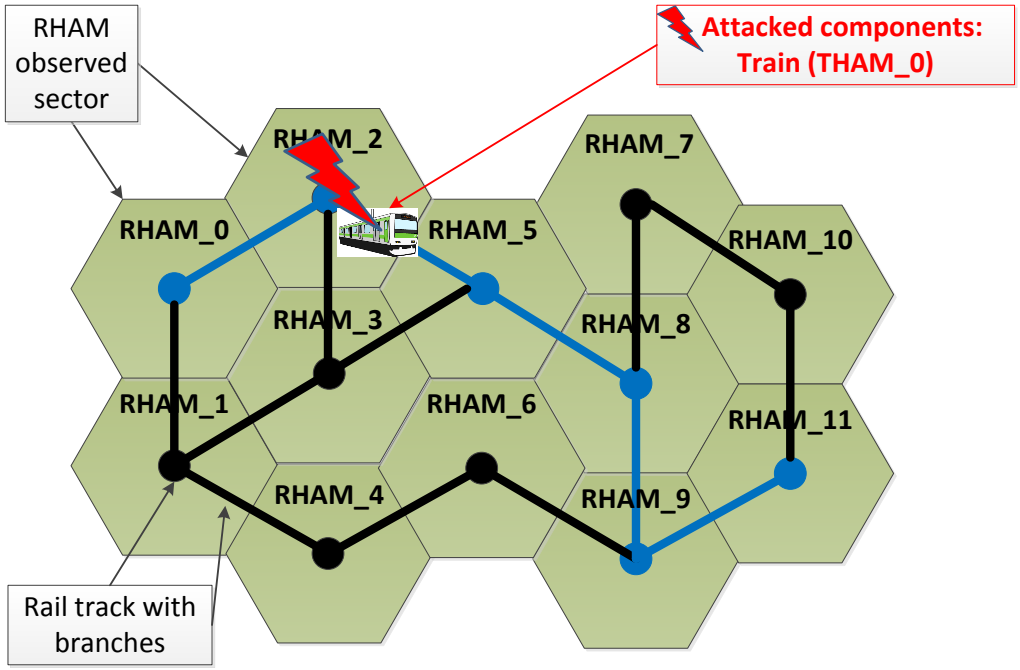


Fig. 32 Simulated Attack Scenario with train under attack

Result description

- In this scenario with a weak EM attack there is no loss of communication between THAM and CHAM
- When an attack occurs the THAM recognizes it after 0.184s
- In a next step the THAM can inform the CHAM about the ongoing attack. So the CHAM receives the information after 0.308s which results from calculating
    - *THAM attack detection time (0.184s) + CHAM attack detection time (0.123s) = 0.308s*
- Because of no communication loss there is no necessity of activating the MBCS on both sides and to switch the communication interface.
- The only measure that is initiated by the CHAM is the broadcast process to inform the other Hams about the ongoing attack. This event happens 0.150 seconds after the attack detection on CHAM side. So the complete process takes
    - *THAM detection time + CHAM detection time + CHAM informs other HAMs time = 0.458s*
- For all other empty events there is no time to be captured
- Conclusion: When the train is attacked and the communication is only disturbed but never lost one can see the reaction behaviour of all components in the detection of the ongoing attack. The attack detection on the train works and the report to CHAM results in a fast information cascade so that after 0.458s the complete railway topology is informed about the attacked train. Because it is a weak attack there is no necessity to switch the MBCS communication interface between the train and CHAM

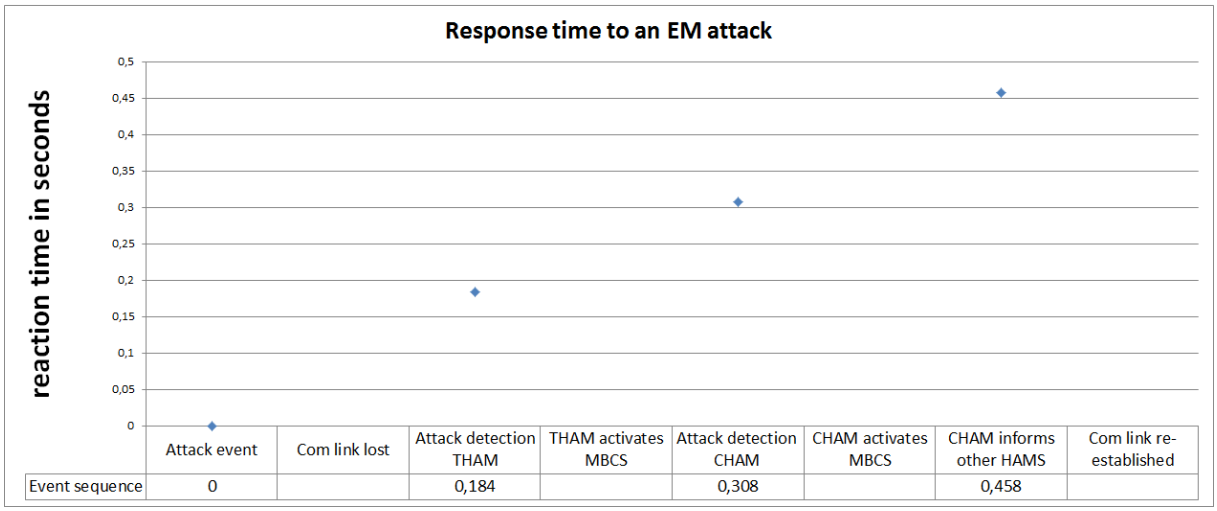


Fig. 33 Response time of the DPS in Base Model 8 (Train under attack) as line with data points

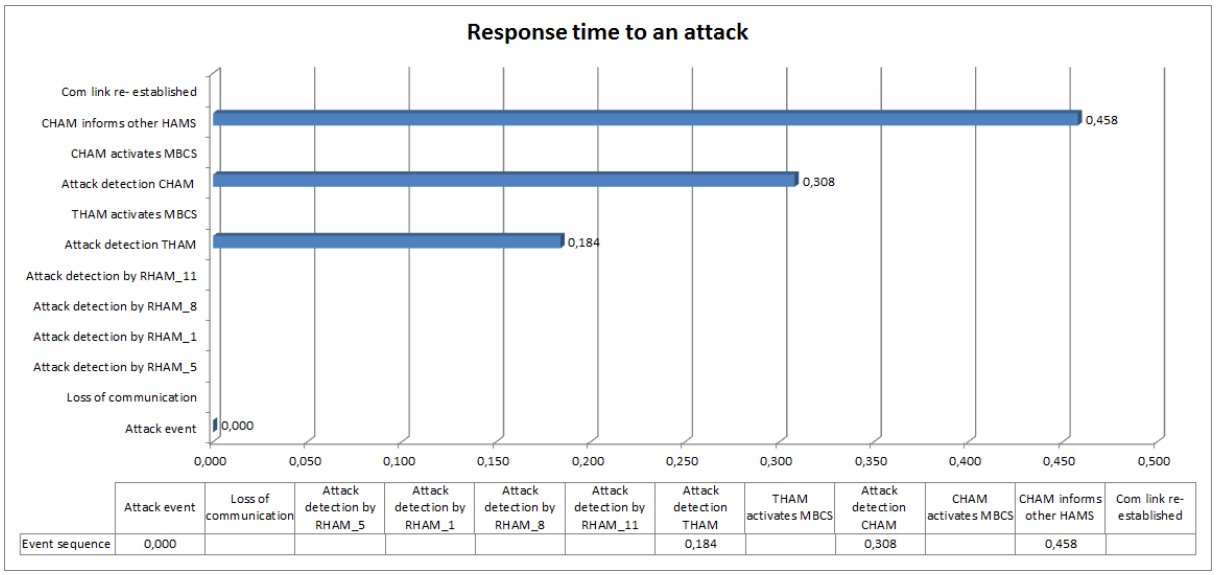


Fig. 34 Response time of the DPS in Base Model 8 (Train under attack) as bar graph

6.2.5SECRET Base Model_11

- Attacked device: RHAM_5 (track health/attack manager)
- Attack severity: high Impact
- Attack duration: 10 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_11 is a completely different scenario in comparison with the previous ones because the focus lies on the railway health/attack manager (RHAM_5). This means that now the jamming attack device is no longer onboard the train but somewhere along the track in the range of RHAM_5.

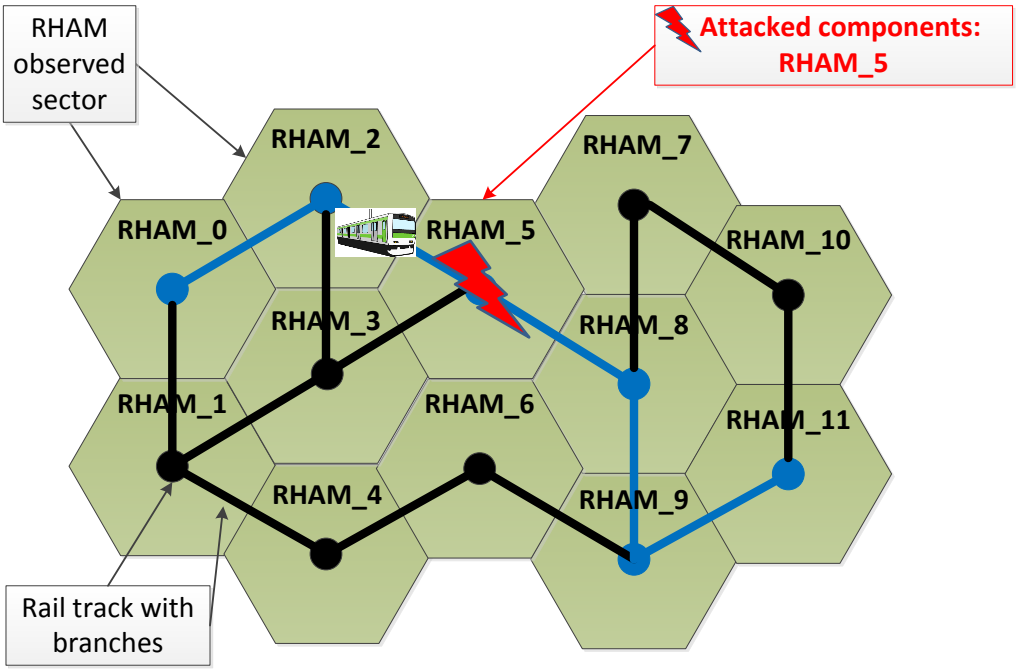


Fig. 35 Simulated Attack Scenario with RHAM_5 sector under attack

Result description

Attacking a railway health/attack manager is a complete different case now. We have to

distinguish three different cases. In most cases the train is not directly affected by the EM attack. Either the train has not yet reached the affected area (case 1) or the train has already passed the affected area (case 3). There is only one case when the train communication is attacked. This is when the train is in RHAM_5 area during RHAM_5 is under attack (case 2).

Case 1 (train before attacked area)

- Keep alive timer does not expire because the train is not involved in the attack while it drives outside of the attacked area
- As long as the train is driving in front of the attacked area there will be no connection loss
- RHAM_5 detects the attack after 0.148s
- Shortly after RHAM_5 detects the attack the CHAM is informed about the ongoing attack by calculating
    - *RHAM detection time (0.148s) + CHAM detection time (0.055s) = 0.198s*
    - The CHAM reacts a little bit faster than in the previous scenarios. Approximately 100ms earlier. This results by the immediate information process initiated by RHAM_5
- The CHAM activates its MBCS 0.052 seconds after the attack detection. So it completely takes
    - *RHAM detection time + CHAM detection time + MBCS activation time = 0.250s*
- In this case THAM_0 will be informed about the ongoing attack by the CHAM. But the train itself does not detect the attack. The train sensors are not affected. So, this process takes
    - *RHAM detection time + CHAM detection time + CHAM informs other HAMs +MBCS activation time = 0.476s*
- The train reacts on the attack later than the CHAM. This results in a complete communication reactivation time of 0.476s + Com re-establishment time (0.134s) = 0.610s
- Conclusion: When only a RHAM is attacked and the train is not involved but is going to enter the attacked area the CHAM has to ensure that the CHAM-train communication won't be lost when the train enters the attacked area of RHAM_5. The new communication interface is active 0.610s after the attack has occurred. The CHAM is also informed a little bit faster than in previous scenarios.

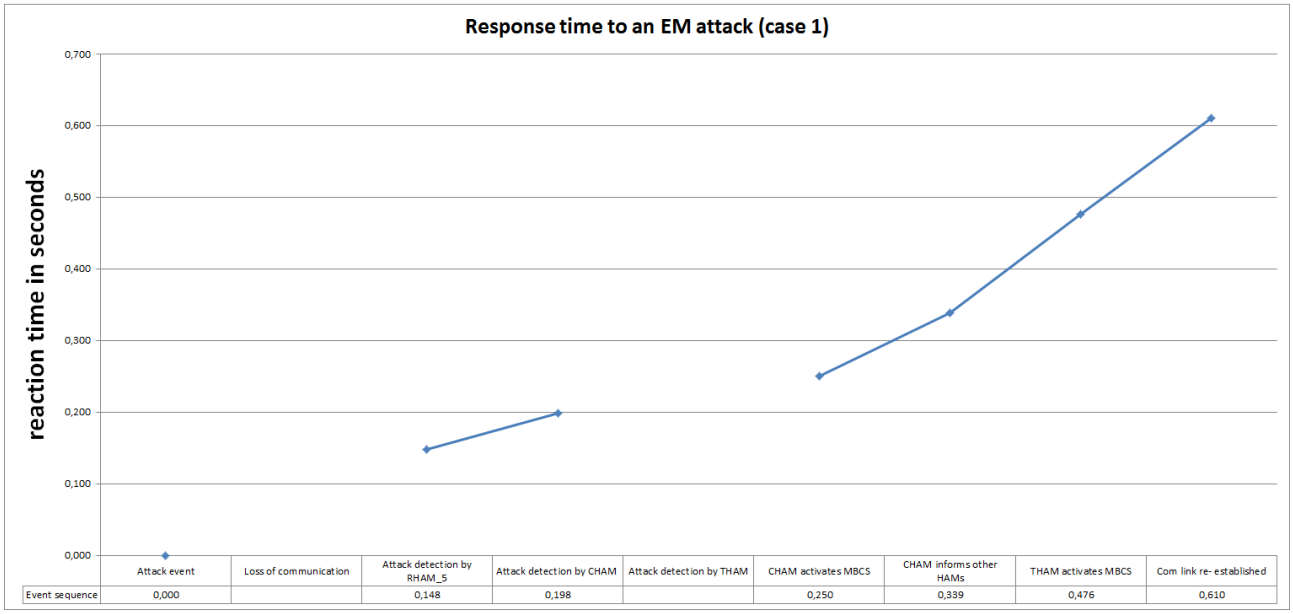


Fig. 36 Response time of the DPS in Base Model 11 (Train under attack) as line with data points

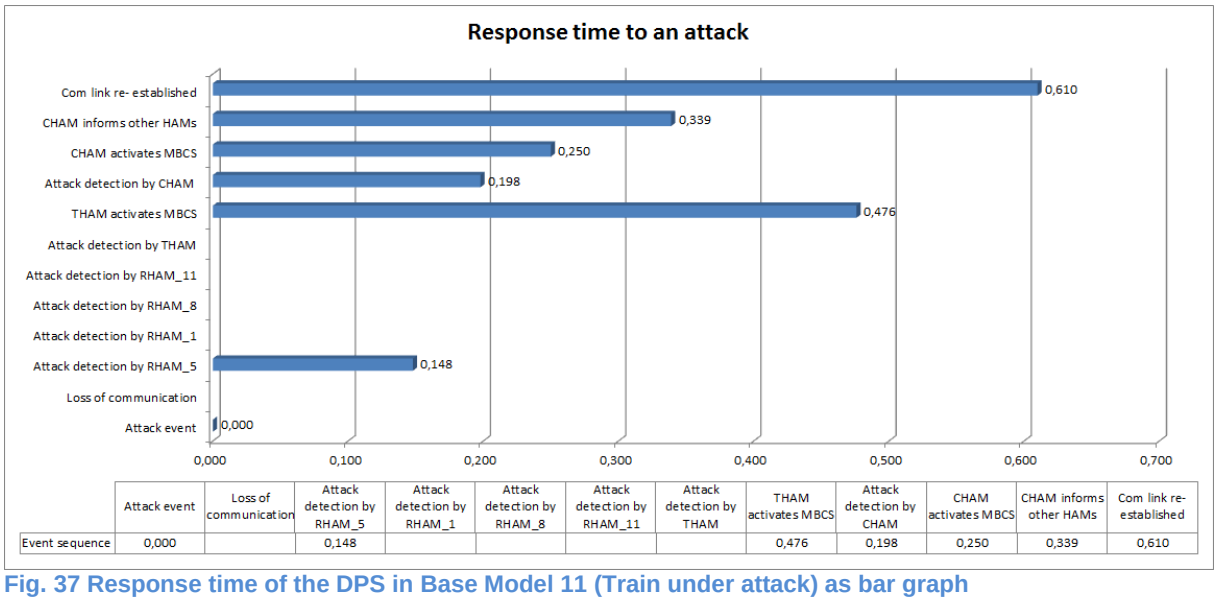
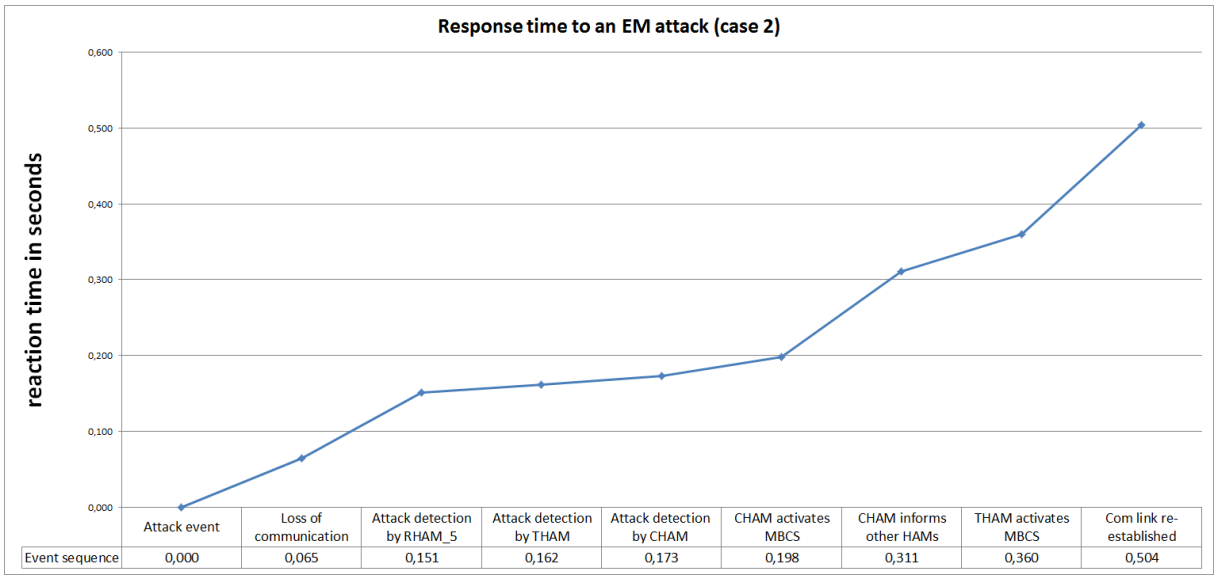


Fig. 37 Response time of the DPS in Base Model 11 (Train under attack) as bar graph

Case 2 (train inside attacked area)

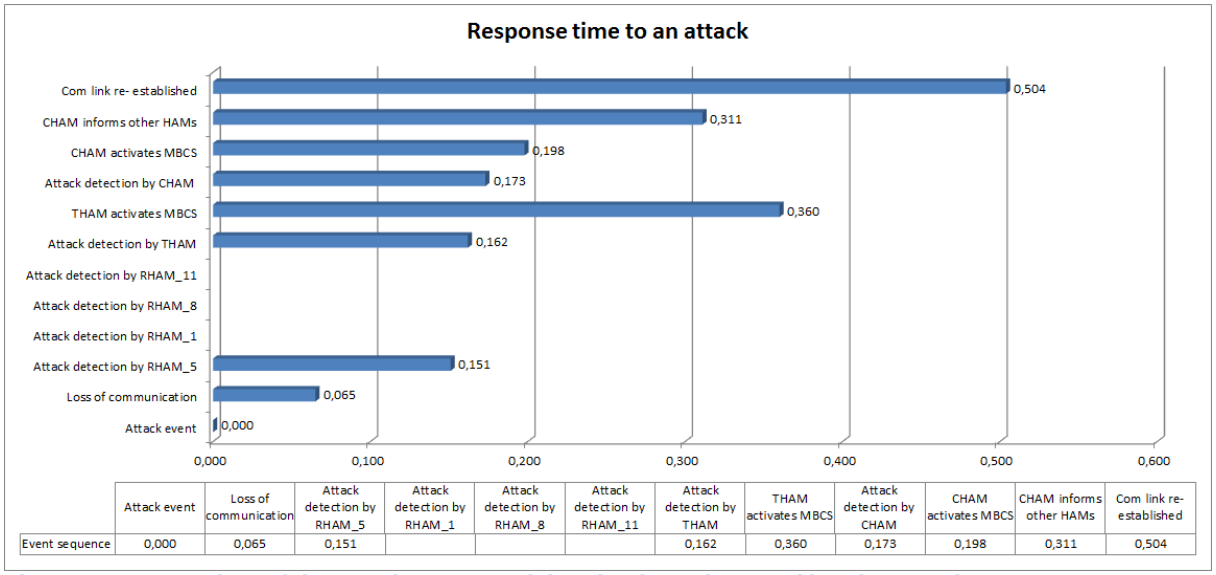
- Here the keep alive timer expires after 5.174s
- The communication is lost 0.065s after the attack moment.
- RHAM_5 detects the attack after 0.151s and
- THAM_0 detects the attack after 0.227s. This shows that they both recognize the attack in the nearly identical time. If the train does not detect the attack because the jamming signal is too weak to be detected then the CHAM would provide the attack information received by the attacked RHAM to the THAM and initiate the communication interface switch.
- Because the communication between train and CHAM is already lost the keep alive timer of the train decreases at CHAM. While the KAT is going to expire and the CHAM is monitoring this progress the CHAM receives the attack information from another source: RHAM_5. The CHAM attack detection time is calculated by

  - *Attack detection by RHAM_5 (0.151) + attack detection by CHAM (0.022) = 0.173s*
- By this reason the CHAM ignores the KAT of the train and immediately reacts on the attack information from RHAM_5. This means that the CHAM is much earlier activating its MBCS than in previous scenarios where the CHAM is waiting for KAT expiration. The calculation is
  - *Attack detection by RHAM_5 (0.151) + attack detection by CHAM (0.022) + MBCS activation time (0.025s) = 0.198s*
- The THAM activates its MBCS 0.133s after the attack moment. So it takes completely
  - Connection loss + Attack detection + MBCS activation = 0.360s
- The CHAM informs other HAMs about the ongoing attack after 0.138s. So the time until all other HAMs know the attack is:
  - *Attack detection by RHAM_5 + Attack detection by CHAM + CHAM informs other HAMs time = 0.311s*
- When we compare these values with previous scenarios, here the THAM is reacting later than the CHAM. So the time to re-establish communication between THAM and CHAM depends on the train reaction time
  - *Connection loss + Attack detected by THAM + MBCS activation + Com link re-establishment time = 0.504s*
- Conclusion: In case 2 the train is already in the area of RHAM_5 when this component will be attacked. This means that the train, although it is not directly attacked, is affected by the influence of the attack device which is somewhere at the track. So, we here have a connection loss between train and CHAM. The CHAM begins to observe the train keep alive timer. But this has no impact on the CHAM reaction time for re-establishing the communication. Because of the wired connection with RHAM_5 it receives the attack information and can immediately switch the communication interface. On the other side, the train also detects the attack and reacts approximately at the same time with the effect that the communication is re-established after 0.504s.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by THAM	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re-established
Event sequence	0,000	0,065	0,151	0,162	0,173	0,198	0,311	0,360	0,504

Fig. 38 Response time of the DPS in Base Model 11 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,065	0,151				0,162	0,360	0,173	0,198	0,311	0,504

Fig. 39 Response time of the DPS in Base Model 11 (Train under attack) as bar graph

Case 3 (train has passed attacked area)

- The attack occurs when the train has already left the affected area of RHAM_5.
- What we could measure is the attack detection time of RHAM_5 after 0.148s. Depending on this follows the attack detection time of the CHAM after further 0.050s and the reporting process of the CHAM to inform all other HAMs in the railway topology.
- Conclusion: Because the train has already passed RHAM_5 on its route there is no necessity for the train to initiate MBCS. The same thing happens on the side of the CHAM. It knows that there is an attack in the topology but the train is not in danger for communication loss at the moment.

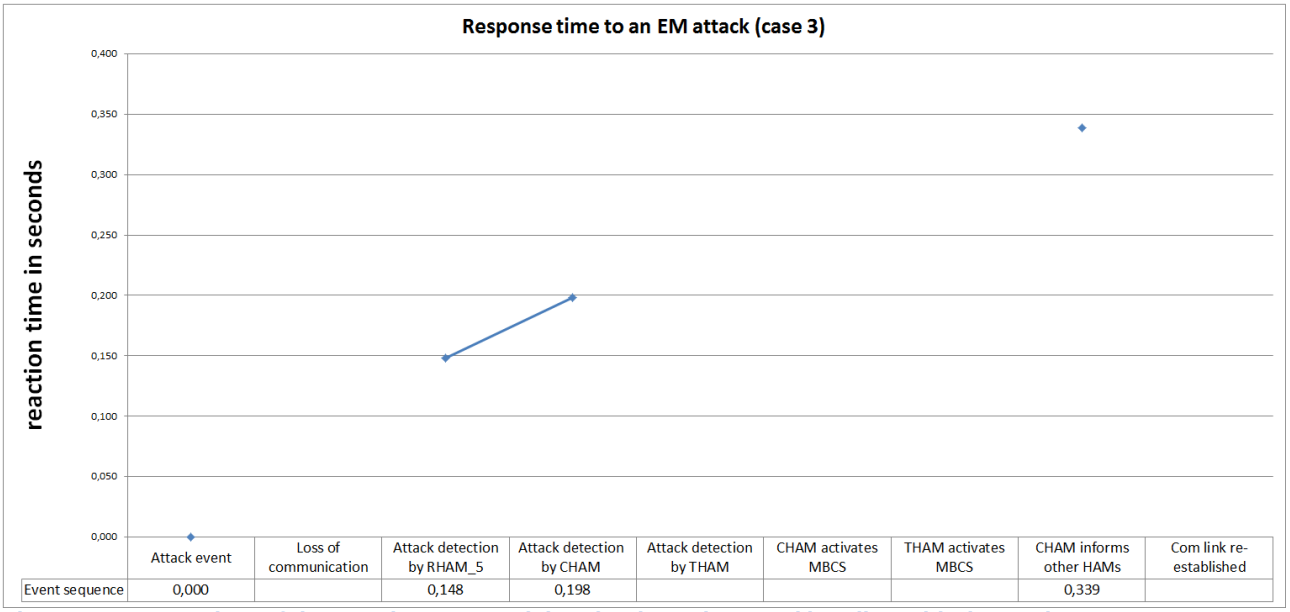


Fig. 40 Response time of the DPS in Base Model 11 (Train under attack) as line with data points

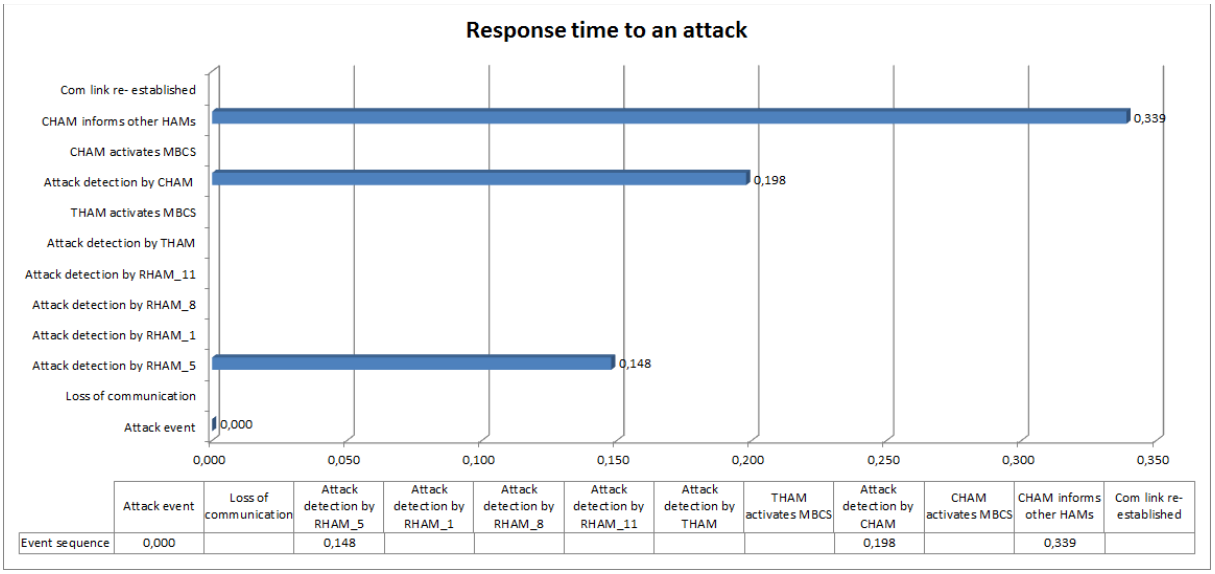


Fig. 41 Response time of the DPS in Base Model 11 (Train under attack) as bar graph

Conclusion in Model 11:

The communication reestablishment in case 1 and 2 is much faster than in previous scenarios where the train is directly involved into the attack. The reason for that is that we here don't have any keep alive timer which delays the re-establishment process. A next fact is that case two is approximately 100ms faster than case 1. This is caused by an additional simulation step in case 1 in comparison to case two. The additional step in case one is the state change report from CHAM to THAM which takes one more simulation cycle. But maybe also in a real system this additional communication step would increase the reaction time by a similar value.

6.2.6 SECRET Base Model_3

- Attacked device: RHAM_5 (track health/attack manager)
- Attack severity: high Impact
- Attack duration: 5 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_3 is similar to Model_11 and analogous to the practice in previous scenarios where we want to compare several scenarios with the RHAM_5 as attacked component considering different attack durations.

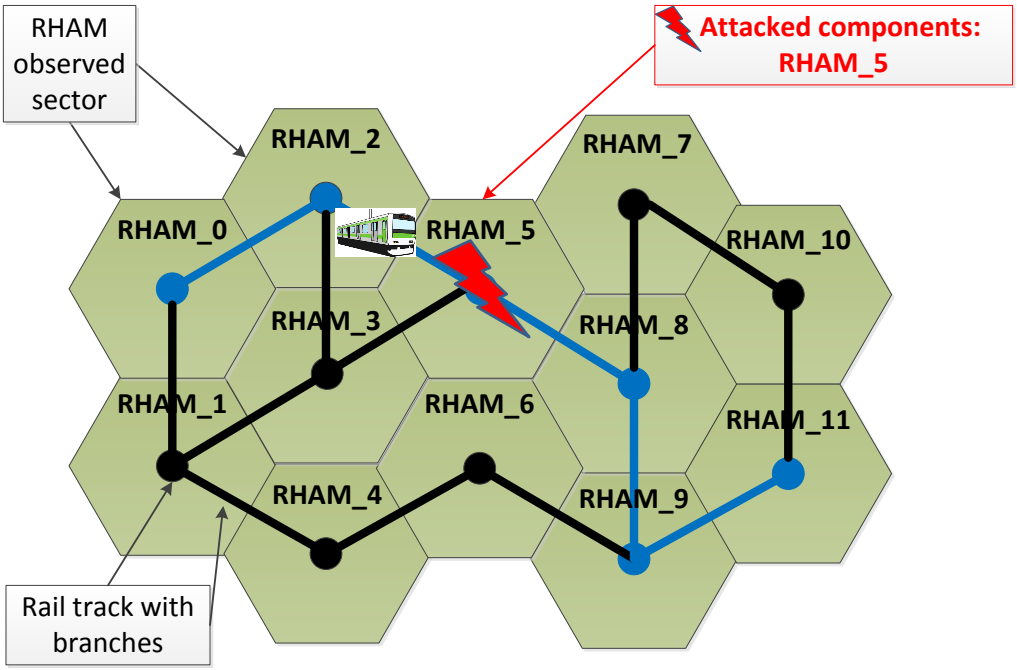


Fig. 42 Simulated attack scenario with RHAM_5 sector under attack

Result description
Case 1 (train before attacked area)

- Keep alive timer does not expire because the train is not involved in the attack while it drives outside of the attacked area
- As long as the train is driving in front of the attacked area there will be no connection loss
- RHAM_5 detects the attack after 0.152s
- Shortly after RHAM_5 detects the attack the CHAM is informed about the ongoing attack by calculating
  - *RHAM detection time (0.152s) + CHAM detection time (0.048s) = 0.200s*
  - The CHAM reacts a little bit faster than in the previous scenarios. Approximately 100ms earlier. This results by the immediate information process initiated by RHAM_5
- The CHAM activates its MBCS 0.050 seconds after the attack detection. So it completely takes
  - *RHAM detection time + CHAM detection time + MBCS activation time = 0.250s*

- In this case THAM_0 will be informed about the ongoing attack by the CHAM. But the train itself does not detect the attack. The train sensors are not affected. So, this process takes
  - *RHAM detection time + CHAM detection time + CHAM informs other HAMs +MBCS activation time = 0.521s*
- The train reacts on the attack later than the CHAM. This results in a complete communication reactivation time of 0.521s + Com re-establishment time (0.164s) = 0.685s

Conclusion: When only a RHAM is attacked and the train is not involved but is going to enter the attacked area the CHAM has to ensure that the CHAM-train communication won't be lost when the train enters the attacked area of RHAM_5. The new communication interface is active 0.685s after the attack has occurred. The CHAM is also informed a little bit faster than in previous scenarios.

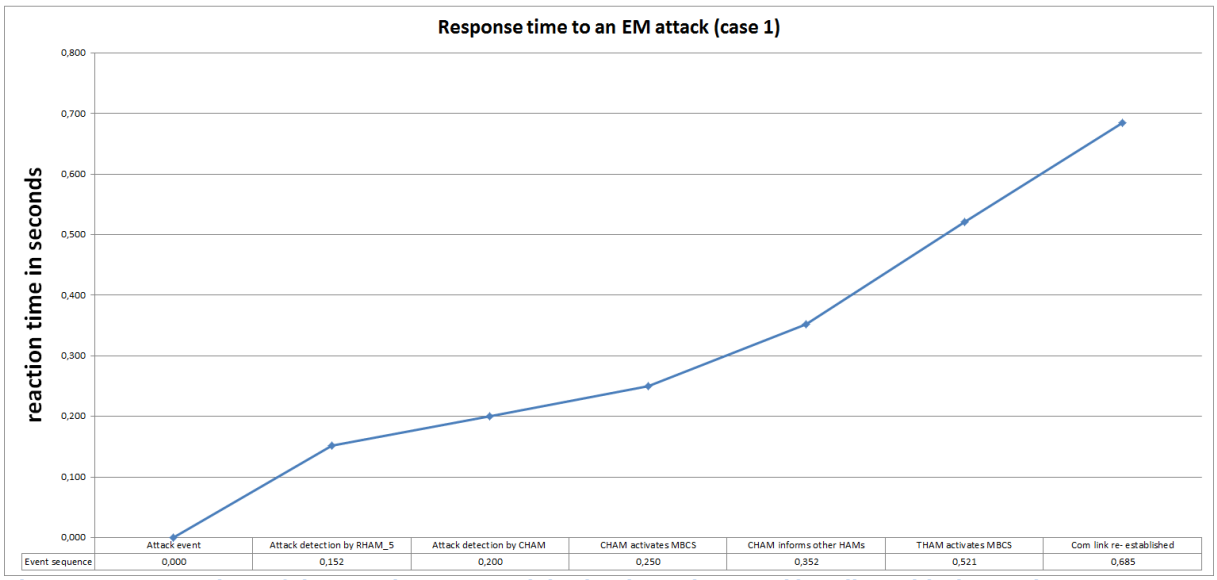


Fig. 43 Response time of the DPS in Base Model 3 (Train under attack) as line with data points

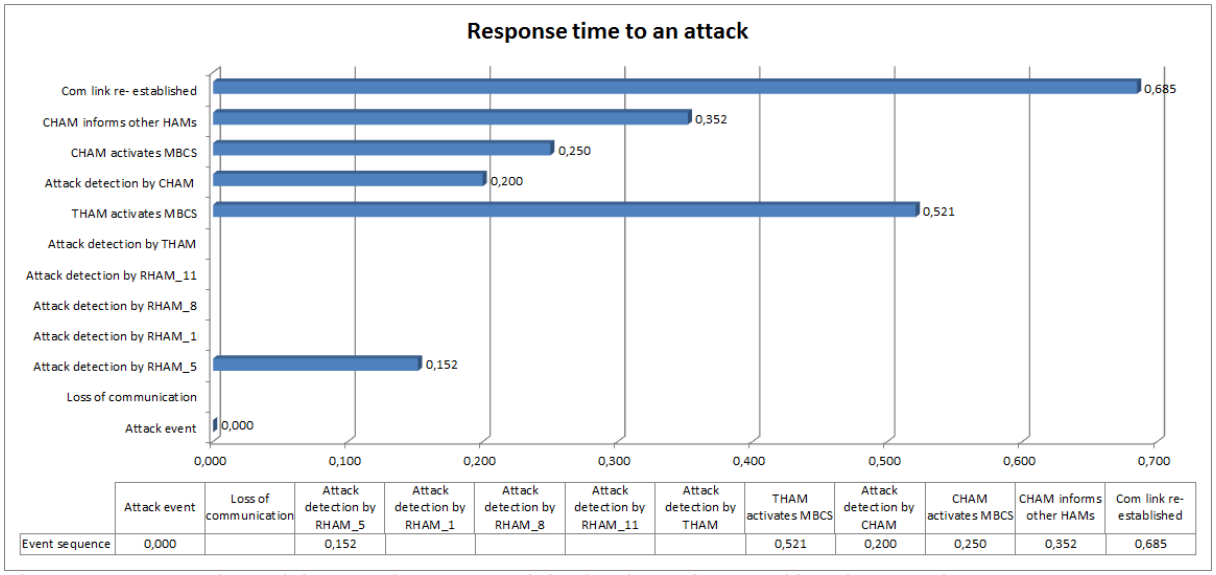
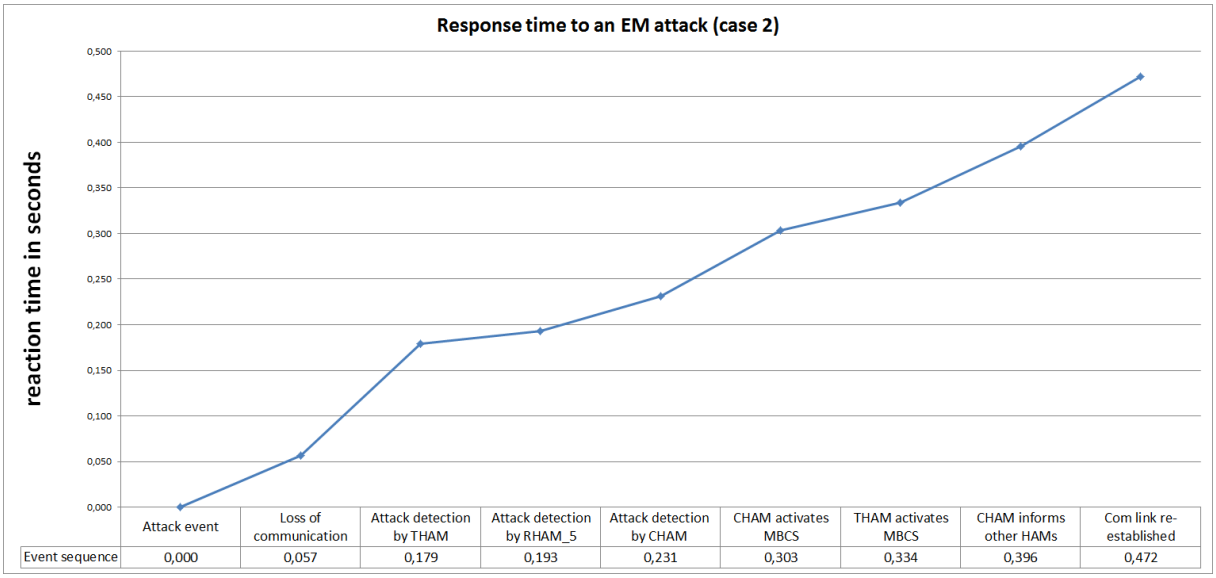


Fig. 44 Response time of the DPS in Base Model 3 (Train under attack) as bar graph

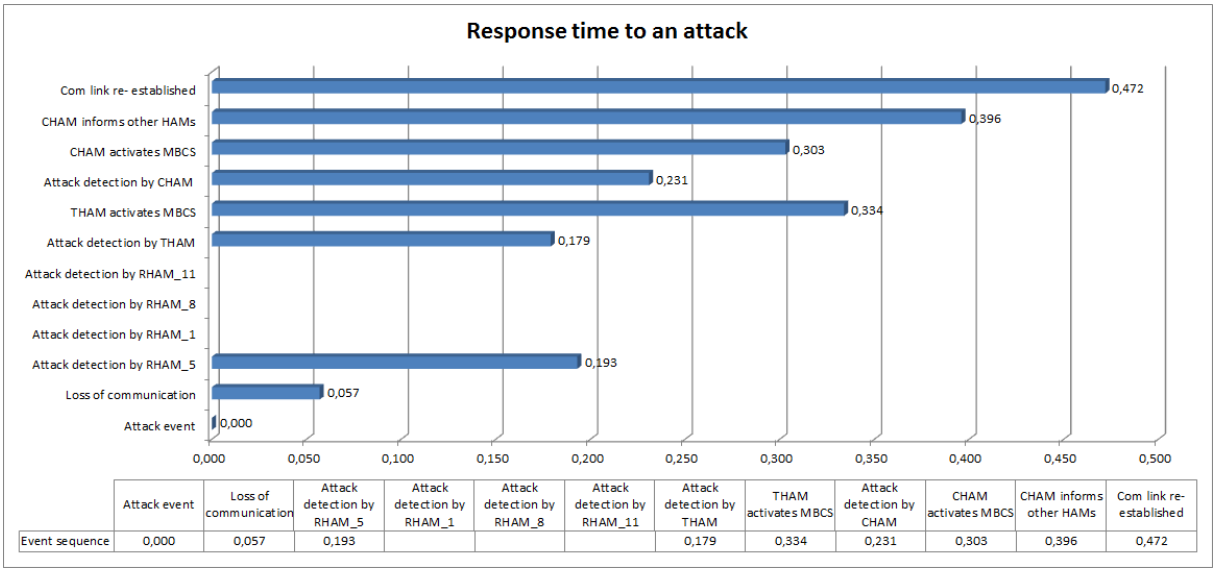
Case 2 (train inside attacked area)

- Here the keep alive timer expires after 5.046
- The communication is lost 0.057s after the attack moment.
- RHAM_5 detects the attack after 0.151s and
- THAM_0 detects the attack after 0.179s. This shows that they both recognize the attack in the nearly identical time.
- Because the communication between train and CHAM is already lost the keep alive timer of the train decreases at CHAM. While the KAT is going to expire and the CHAM is monitoring this progress the CHAM receives the attack information from another source: RHAM_5. The CHAM attack detection time is calculated by
    - *Attack detection by RHAM_5 (0.151) + attack detection by CHAM (0.022) = 0.173s*
- By this reason the CHAM ignores the KAT of the train and immediately reacts on the attack information from RHAM_5. This means that the CHAM is much earlier activating its MBCS than in previous scenarios where the CHAM is waiting for KAT expiration. The calculation is
    - *Attack detection by RHAM_5 (0.193) + attack detection by CHAM (0.038) + MBCS activation time (0.072s) = 0.303s*
- The THAM activates its MBCS 0.155s after the attack moment. So it takes completely
    - Connection loss + Attack detection + MBCS activation = 0.360s
- The CHAM informs other HAMs about the ongoing attack after 0.165s. So the time until all other HAMs know the attack is:
    - *Attack detection by RHAM_5 + Attack detection by CHAM + CHAM informs other HAMs time = 0.396s*
- When we compare these values with previous scenarios, here the THAM is reacting later than the CHAM. So the time to re-establish communication between THAM and CHAM depends on the train reaction time
    - *Connection loss + Attack detected by THAM + MBCS activation + Com link re-establishment time = 0.472s*
- Conclusion: In case 2 the train is already in the area of RHAM_5 when this component will be attacked. This means that the train, although it is not directly attacked, is affected by the influence of the attack device which is somewhere at the track. So, we here have a connection loss between train and CHAM. The CHAM begins to observe the train keep alive timer. But this has no impact on the CHAM reaction time for re-establishing the communication. Because of the wired connection with RHAM_5 it receives the attack information and can immediately switch the communication interface. On the other side, the train also detects the attack and reacts approximately at the same time with the effect that the communication is re-established after 0.472s.



	Attack event	Loss of communication	Attack detection by THAM	Attack detection by RHAM_5	Attack detection by CHAM	CHAM activates MBCS	THAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,057	0,179	0,193	0,231	0,303	0,334	0,396	0,472

Fig. 45 Response time of the DPS in Base Model 3 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,057	0,193				0,179	0,334	0,231	0,303	0,396	0,472

Fig. 46 Response time of the DPS in Base Model 3 (Train under attack) as bar graph

Case 3 (train has passed attacked area)

- The attack occurs when the train has already left the affected area of RHAM_5.
- What we could measure is the attack detection time of RHAM_5 after 0.152s. Depending on this follows the attack detection time of the CHAM after further 0.048s and the reporting process of the CHAM to inform all other HAMs in the railway topology.
- Conclusion: Because the train has already passed RHAM_5 on its route there is no necessity for the train to initiate MBCS. The same thing happens on the side of the CHAM. It knows that there is an attack in the topology but the train is not in danger for communication loss at the moment.

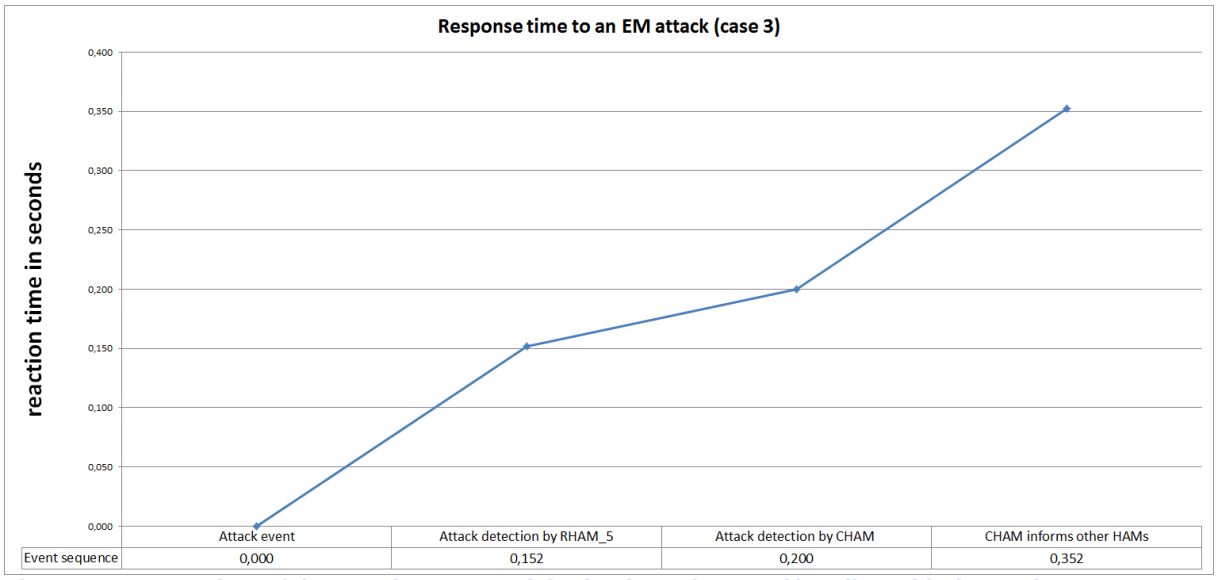


Fig. 47 Response time of the DPS in Base Model 3 (Train under attack) as line with data points

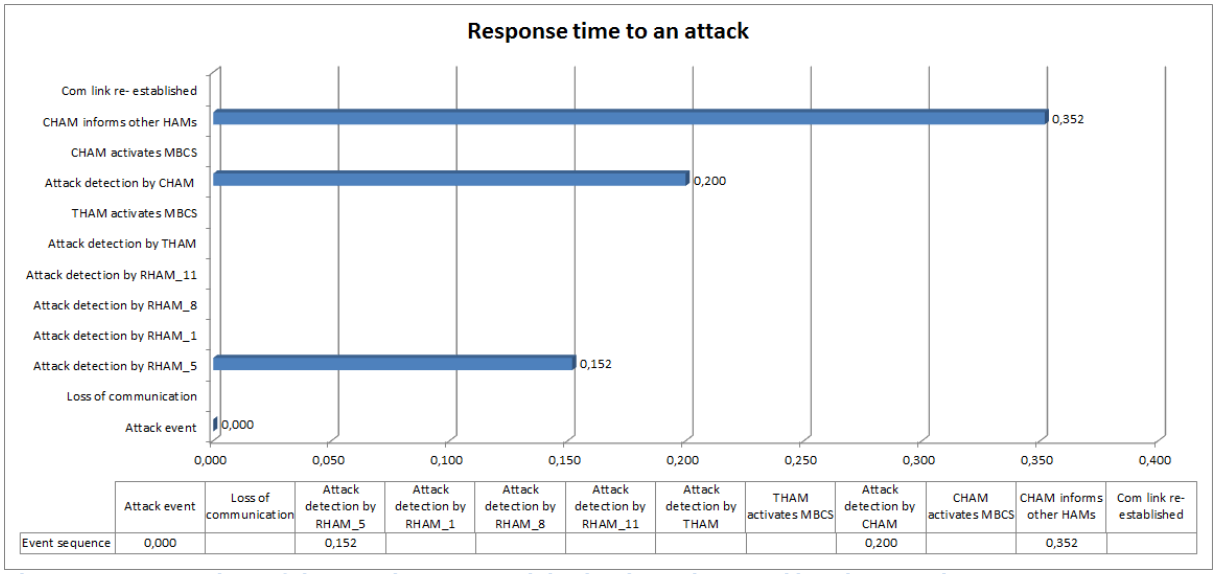


Fig. 48 Response time of the DPS in Base Model 3 (Train under attack) as bar graph

6.2.7SECRET Base Model_2

- Attacked device: RHAM_5 (track health/attack manager)
- Attack severity: high Impact
- Attack duration: 2 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_2 fits to Model_11 and Model_3 because the start configuration only distinguishes in much shorter attack duration.

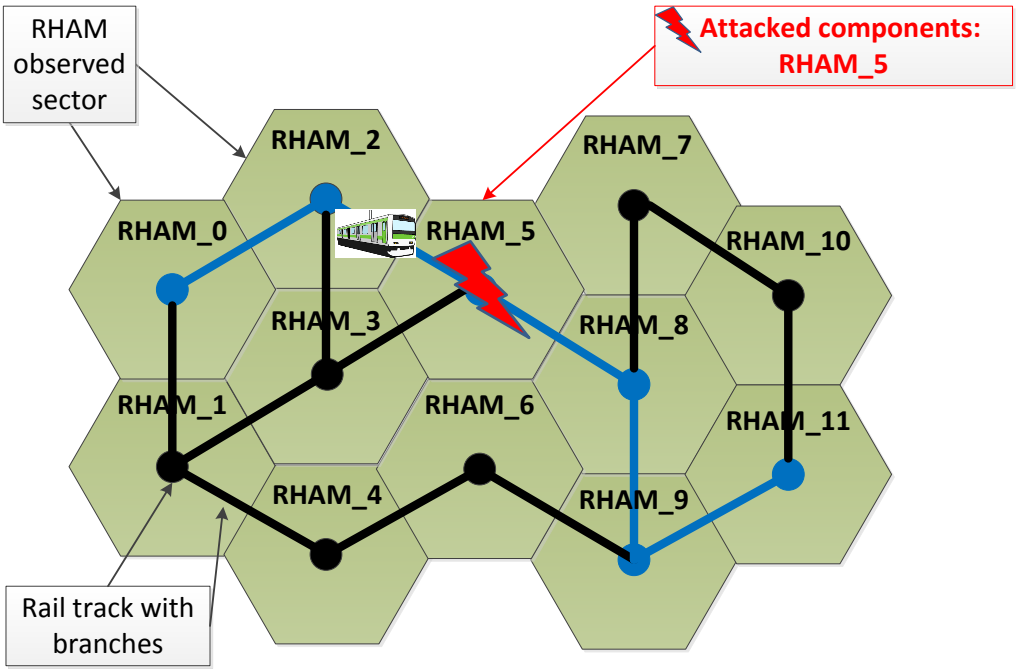


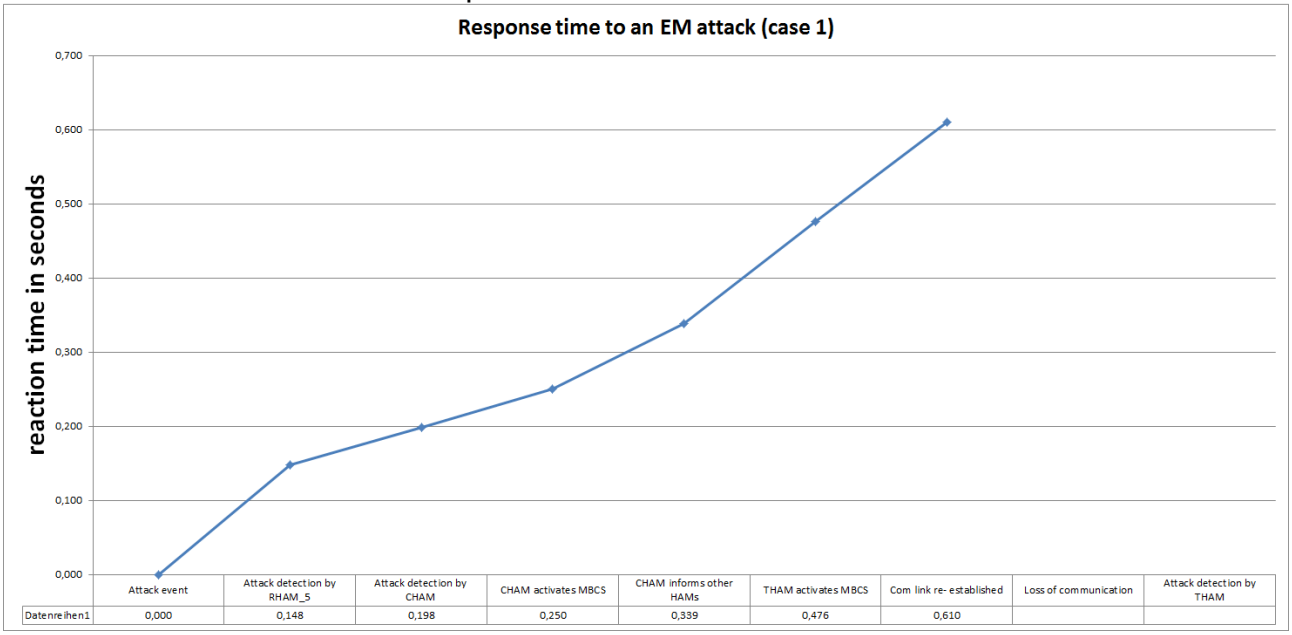
Fig. 49 Simulated attack scenario with RHAM_5 sector under attack

Result description
Case 1 (train before attacked area)

- Keep alive timer does not expire because the train is not involved in the attack while it drives outside of the attacked area
- As long as the train is driving in front of the attacked area there will be no connection loss
- RHAM_5 detects the attack after 0.148s
- Shortly after RHAM_5 detects the attack the CHAM is informed about the ongoing attack by calculating
  - *RHAM detection time (0.148s) + CHAM detection time (0.050s) = 0.198s*
  - The CHAM reacts a little bit faster than in the previous scenarios. Approximately 100ms earlier. This results by the immediate information process initiated by RHAM_5
- The CHAM activates its MBCS 0.052 seconds after the attack detection. So it completely takes
  - *RHAM detection time + CHAM detection time + MBCS activation time = 0.250s*

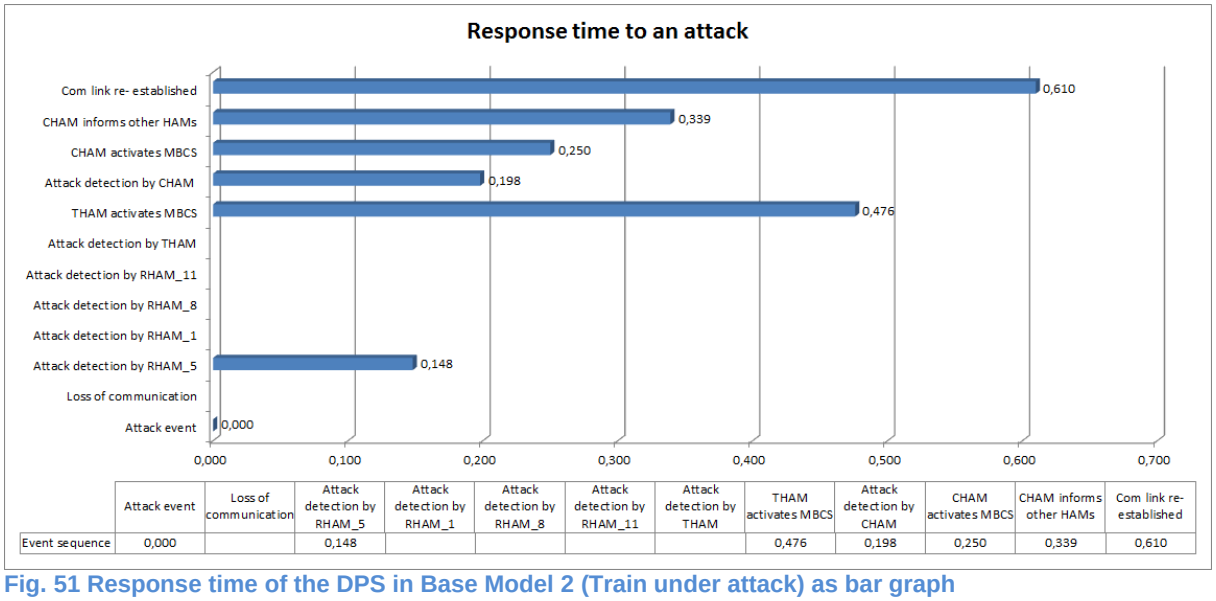
- In this case THAM_0 will be informed about the ongoing attack by the CHAM. But the train itself does not detect the attack. The train sensors are not affected. So, this process takes
  - *RHAM detection time + CHAM detection time + CHAM informs other HAMs +MBCS activation time = 0.476s*
- The train reacts on the attack later than the CHAM. This results in a complete communication reactivation time of 0.476s + Com re-establishment time (0.144s) = 0.610s

Conclusion: When only a RHAM is attacked and the train is not involved but is going to enter the attacked area the CHAM has to ensure that the CHAM-train communication won't be lost when the train enters the attacked area of RHAM_5. The new communication interface is active 0.610s after the attack has occurred. The CHAM is also informed a little bit faster than in previous scenarios.



	Attack event	Attack detection by RHAM_5	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re- established	Loss of communication	Attack detection by THAM
Datenreihen1	0,000	0,148	0,198	0,250	0,339	0,476	0,610		

Fig. 50 Response time of the DPS in Base Model 2 (Train under attack) as line with data points

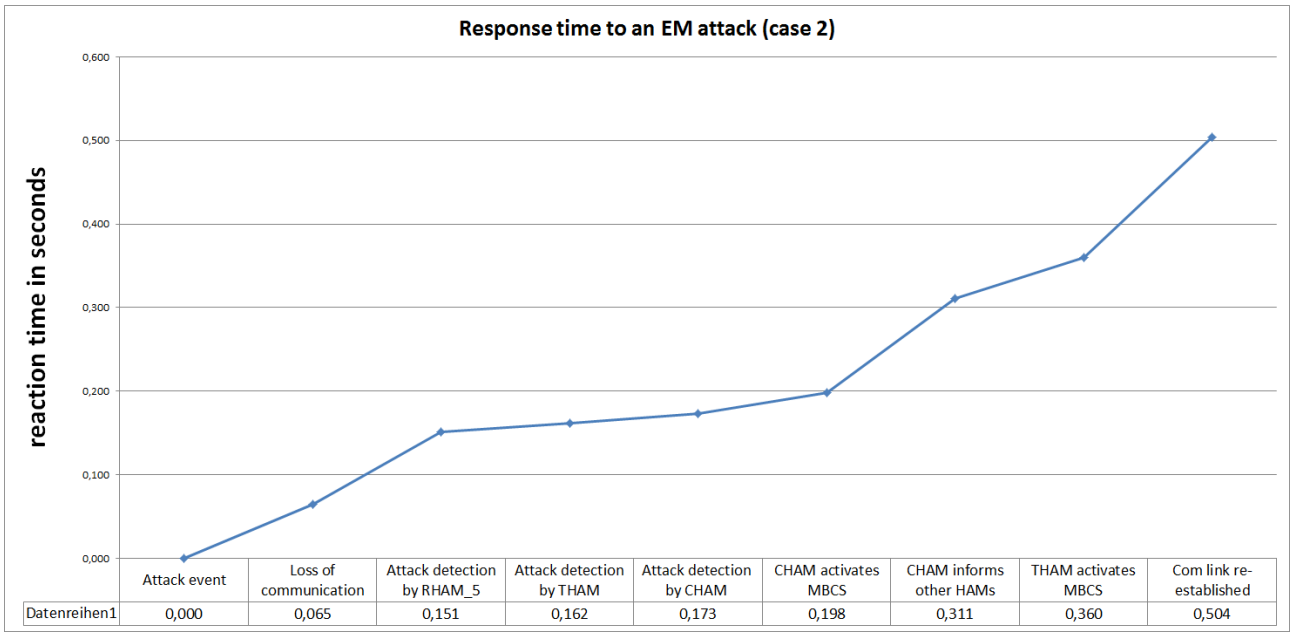


	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000		0,148					0,476	0,198	0,250	0,339	0,610

Fig. 51 Response time of the DPS in Base Model 2 (Train under attack) as bar graph

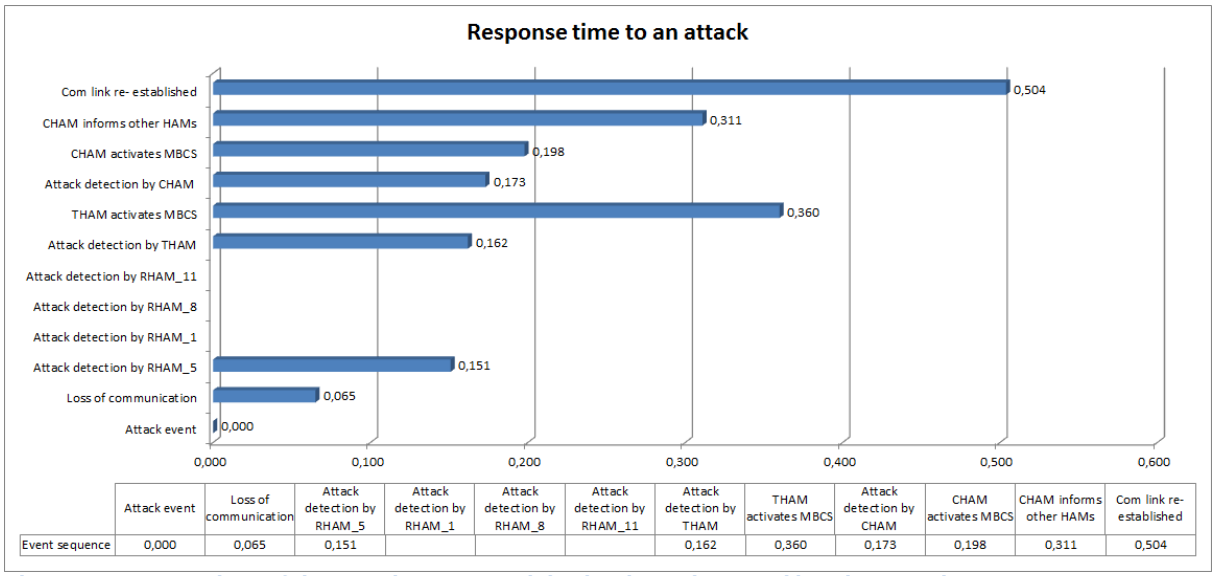
Case 2 (train inside attacked area)

- Here the keep alive timer expires after 5.021s
- The communication is lost 0.048s after the attack moment.
- RHAM_5 detects the attack after 0.138s and
- THAM_0 detects the attack after 0.162s. This shows that they both recognize the attack in the nearly identical time.
- Because the communication between train and CHAM is already lost the keep alive timer of the train decreases at CHAM. While the KAT is going to expire and the CHAM is monitoring this progress the CHAM receives the attack information from another source: RHAM_5. The CHAM attack detection time is calculated by
    - *Attack detection by RHAM_5 (0.151) + attack detection by CHAM (0.022) = 0.173s*
- By this reason the CHAM ignores the KAT of the train and immediately reacts on the attack information from RHAM_5. This means that the CHAM is much earlier activating its MBCS than in previous scenarios where the CHAM is waiting for KAT expiration. The calculation is
    - *Attack detection by RHAM_5 (0.151) + attack detection by CHAM (0.022) + MBCS activation time (0.025s) = 0.198s*
- The THAM activates its MBCS 0.133s after the attack moment. So it takes completely
    - Connection loss + Attack detection + MBCS activation = 0.360s
- The CHAM informs other HAMs about the ongoing attack after 0.138s. So the time until all other HAMs know the attack is:
    - *Attack detection by RHAM_5 + Attack detection by CHAM + CHAM informs other HAMs time = 0.311s*
- When we compare these values with previous scenarios, here the THAM is reacting later than the CHAM. So the time to re-establish communication between THAM and CHAM depends on the train reaction time
    - *Connection loss + Attack detected by THAM + MBCS activation + Com link re-establishment time = 0.504s*
- Conclusion: In case 2 the train is already in the area of RHAM_5 when this component will be attacked. This means that the train, although it is not directly attacked, is affected by the influence of the attack device which is somewhere at the track. So, we here have a connection loss between train and CHAM. The CHAM begins to observe the train keep alive timer. But this has no impact on the CHAM reaction time for re-establishing the communication. Because of the wired connection with RHAM_5 it receives the attack information and can immediately switch the communication interface. On the other side, the train also detects the attack and reacts approximately at the same time with the effect that the communication is re-established after 0.504s.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by THAM	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re-established
Datenreihen1	0,000	0,065	0,151	0,162	0,173	0,198	0,311	0,360	0,504

Fig. 52 Response time of the DPS in Base Model 2 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,065	0,151				0,162	0,360	0,173	0,198	0,311	0,504

Fig. 53 Response time of the DPS in Base Model 2 (Train under attack) as bar graph

Case 3 (train has passed attacked area)

- The attack occurs when the train has already left the affected area of RHAM_5.
- What we could measure is the attack detection time of RHAM_5 after 0.148s. Depending on this follows the attack detection time of the CHAM after further 0.050s and the reporting process of the CHAM to inform all other HAMs in the railway topology.
- Conclusion: Because the train has already passed RHAM_5 on its route there is no necessity for the train to initiate MBCS. The same thing happens on the side of the CHAM. It knows that there is an attack in the topology but the train is not in danger for communication loss at the moment.

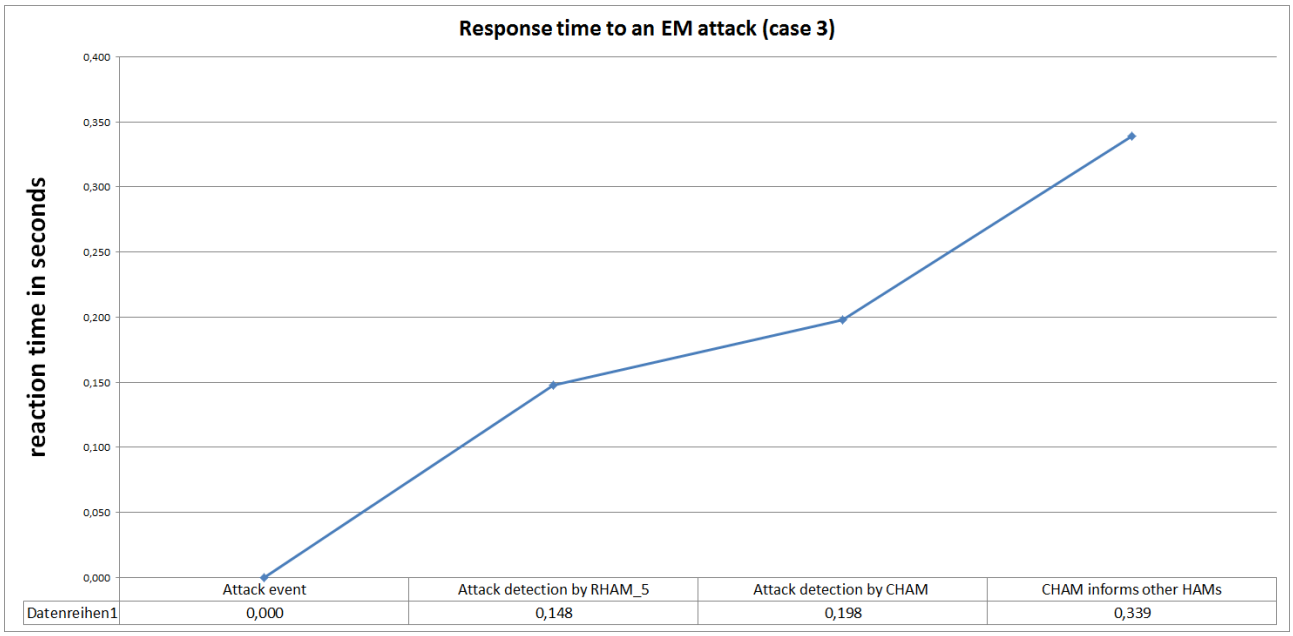


Fig. 54 Response time of the DPS in Base Model 2 (Train under attack) as line with data points

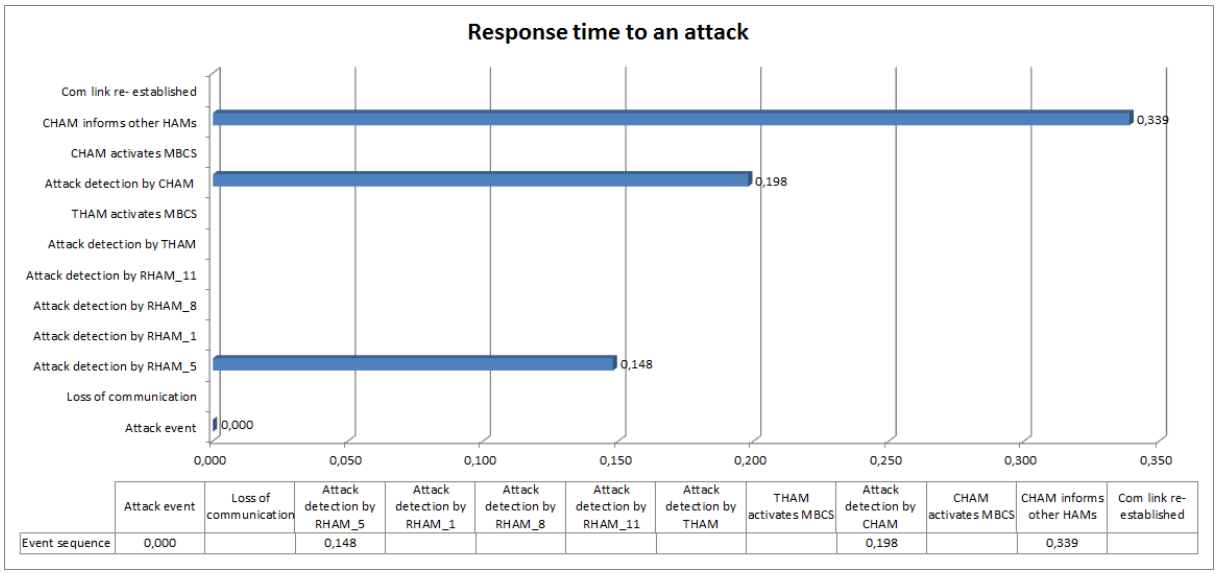


Fig. 55 Response time of the DPS in Base Model 2 (Train under attack) as bar graph

6.2.8SECRET Base Model_4

- Attacked device: RHAM_5 (track health/attack manager)
- Attack severity: medium Impact
- Attack duration: 5 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_4 is similar to the three previous scenarios where RHAM_5 is attacked. Here we modified the attack severity to a medium impact and simulate 5 seconds attack events.

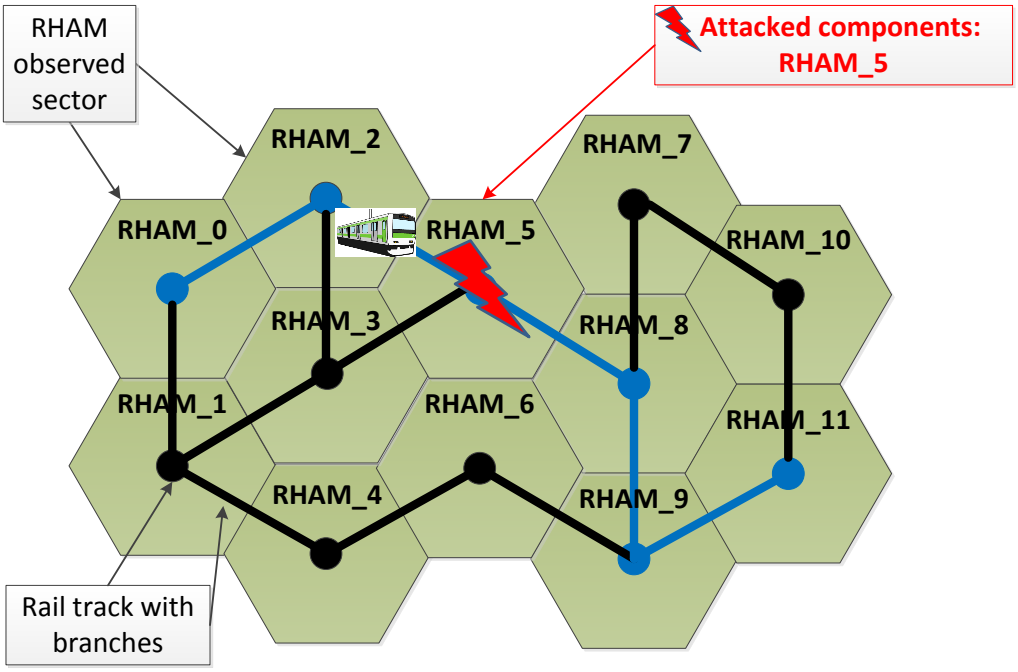


Fig. 56 Simulated attack scenario with RHAM_5 sector under attack

Result description
Case 1 (train before attacked area)

- Keep alive timer does not expire because the train is not involved in the attack while it drives outside of the attacked area
- As long as the train is driving in front of the attacked area there will be no connection loss
- RHAM_5 detects the attack after 0.146s
- Shortly after RHAM_5 detects the attack the CHAM is informed about the ongoing attack by calculating
  - *RHAM detection time (0.146s) + CHAM detection time (0.045s) = 0.191s*
  - The CHAM reacts a little bit faster than in the previous scenarios. Approximately 100ms earlier. This results by the immediate information process initiated by RHAM_5
- The CHAM activates its MBCS 0.046 seconds after the attack detection. So it completely takes
  - *RHAM detection time + CHAM detection time + MBCS activation time = 0.266s*

- In this case THAM_0 will be informed about the ongoing attack by the CHAM. But the train itself does not detect the attack. The train sensors are not affected. So, this process takes
  - *RHAM detection time + CHAM detection time + CHAM informs other HAMs +MBCS activation time = 0.481s*
- The train reacts on the attack later than the CHAM. This results in a complete communication reactivation time of 0.481s + Com re-establishment time (0.150s) = 0.631s

Conclusion: When only a RHAM is attacked and the train is not involved but is going to enter the attacked area the CHAM has to ensure that the CHAM-train communication won't be lost when the train enters the attacked area of RHAM_5. The new communication interface is active 0.631s after the attack has occurred. The CHAM is also informed a little bit faster than in previous scenarios.

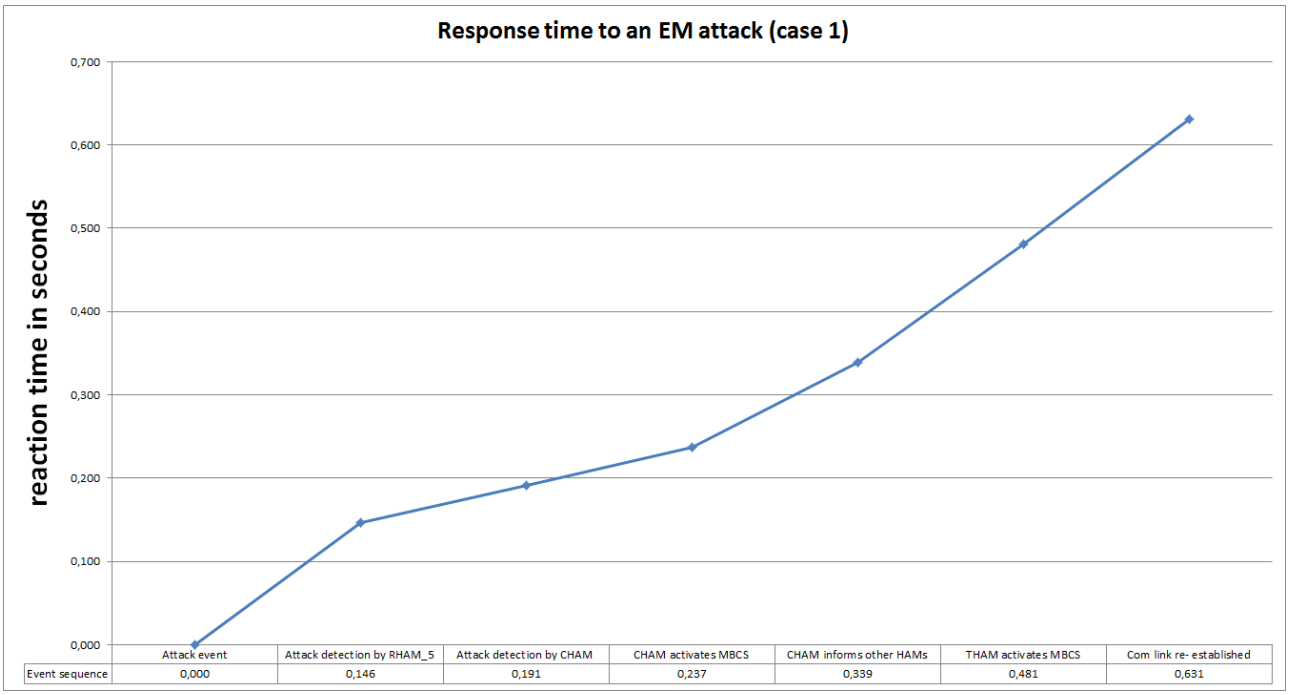
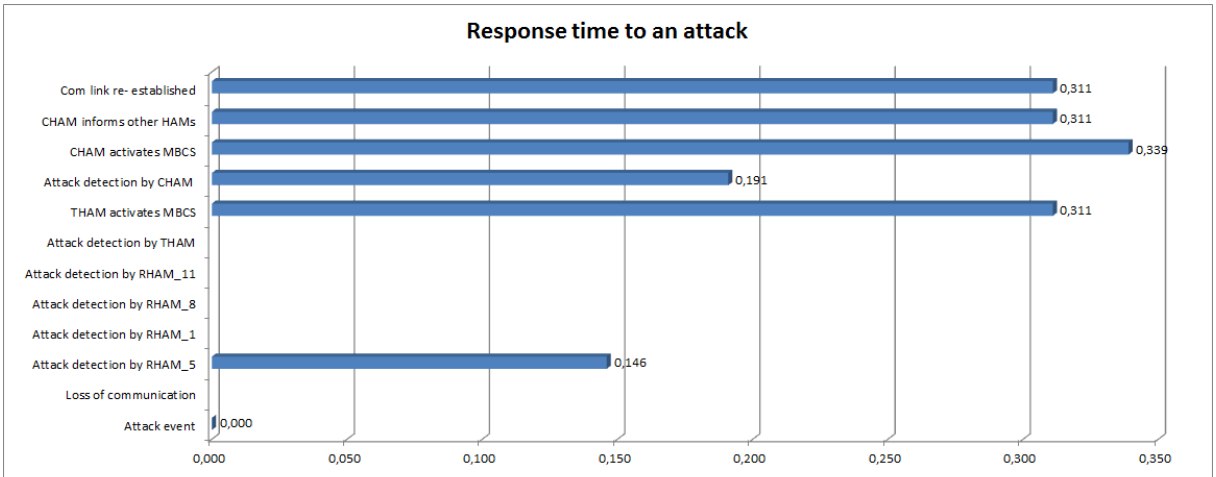


Fig. 57 Response time of the DPS in Base Model 5 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000		0,146					0,311	0,191	0,339	0,311	0,311

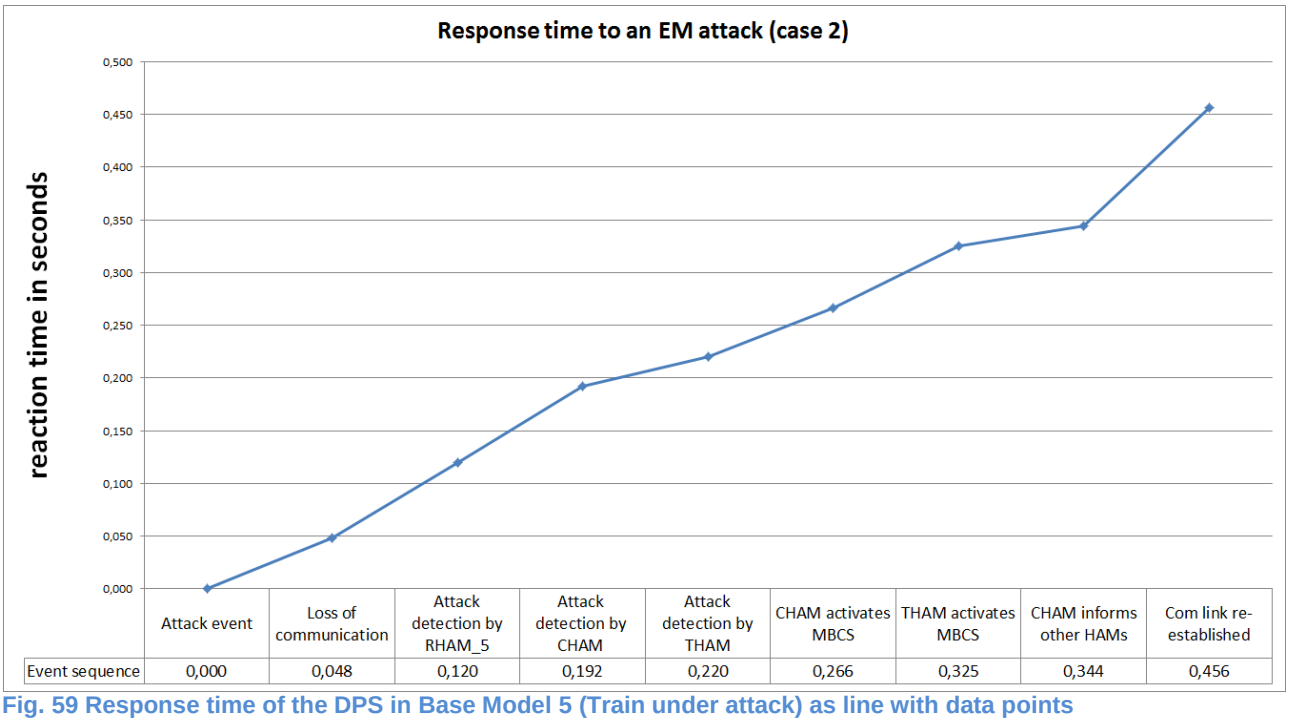
Fig. 58 Response time of the DPS in Base Model 5 (Train under attack) as bar graph

Case 2 (train inside attacked area)

- Here the keep alive timer expires after 5.045s
- The communication is lost 0.048s after the attack moment.
- RHAM_5 detects the attack after 0.120s and
- THAM_0 detects the attack after 0.220s. This shows that they both recognize the attack in the nearly identical time.
- Because the communication between train and CHAM is already lost the keep alive timer of the train decreases at CHAM. While the KAT is going to expire and the CHAM is monitoring this progress the CHAM receives the attack information from another source: RHAM_5. The CHAM attack detection time is calculated by
  - *Attack detection by RHAM_5 (0.120) + attack detection by CHAM (0.072) = 0.192s*
- By this reason the CHAM ignores the KAT of the train and immediately reacts on the attack information from RHAM_5. This means that the CHAM is much earlier activating its MBCS than in previous scenarios where the CHAM is waiting for KAT expiration. The calculation is
  - *Attack detection by RHAM_5 (0.120) + attack detection by CHAM (0.072) + MBCS activation time (0.074s) = 0.266s*
- The THAM activates its MBCS 0.142s after the attack moment. So it takes completely
  - Connection loss + Attack detection + MBCS activation = 0.325s
- The CHAM informs other HAMs about the ongoing attack after 0.149s. So the time until all other HAMs know the attack is:
  - *Attack detection by RHAM_5 + Attack detection by CHAM + CHAM informs other HAMs time = 0.344s*
- When we compare these values with previous scenarios, here the THAM is reacting later than the CHAM. So the time to re-establish communication between THAM and CHAM depends on the train reaction time

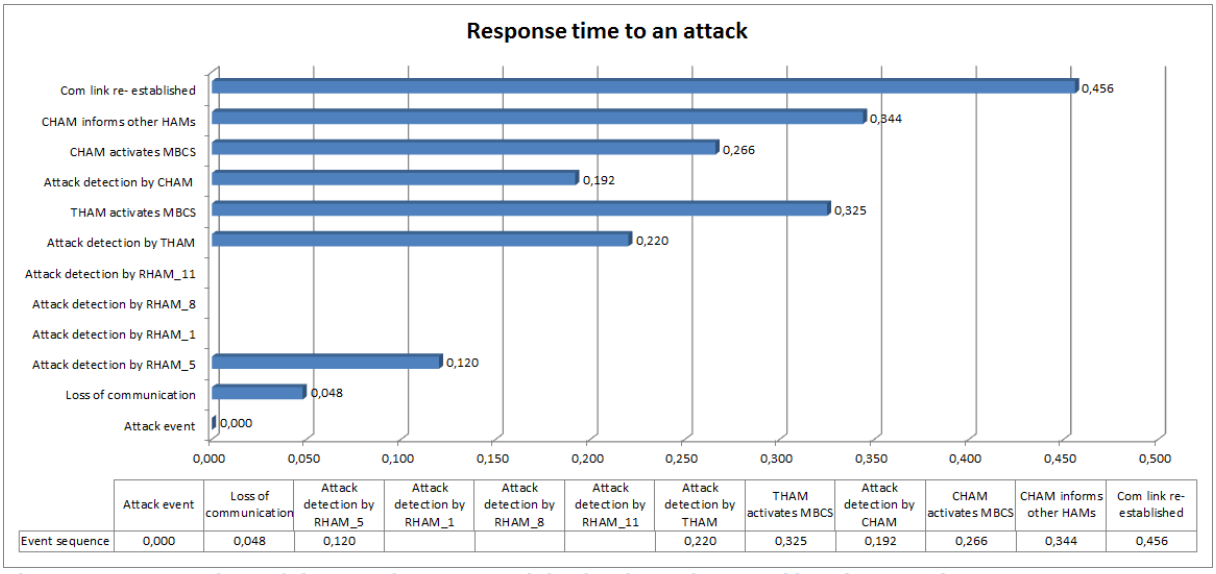
- *Connection loss + Attack detected by THAM + MBCS activation + Com link re-establishment time = 0.456s*

- Conclusion: In case 2 the train is already in the area of RHAM_5 when this component will be attacked. This means that the train, although it is not directly attacked, is affected by the influence of the attack device which is somewhere at the track. So, we here have a connection loss between train and CHAM. The CHAM begins to observe the train keep alive timer. But this has no impact on the CHAM reaction time for re-establishing the communication. Because of the wired connection with RHAM_5 it receives the attack information and can immediately switch the communication interface. On the other side, the train also detects the attack and reacts approximately at the same time with the effect that the communication is re-established after 0.456s.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by CHAM	Attack detection by THAM	CHAM activates MBCS	THAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,048	0,120	0,192	0,220	0,266	0,325	0,344	0,456

Fig. 59 Response time of the DPS in Base Model 5 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,048	0,120				0,220	0,325	0,192	0,266	0,344	0,456

Fig. 60 Response time of the DPS in Base Model 5 (Train under attack) as bar graph

Case 3 (train has passed attacked area)

- The attack occurs when the train has already left the affected area of RHAM_5.
- What we could measure is the attack detection time of RHAM_5 after 0.146s. Depending on this follows the attack detection time of the CHAM after further 0.045s and the reporting process of the CHAM to inform all other HAMs in the railway topology.
- Conclusion: Because the train has already passed RHAM_5 on its route there is no necessity for the train to initiate MBCS. The same thing happens on the side of the CHAM. It knows that there is an attack in the topology but the train is not in danger for communication loss at the moment.

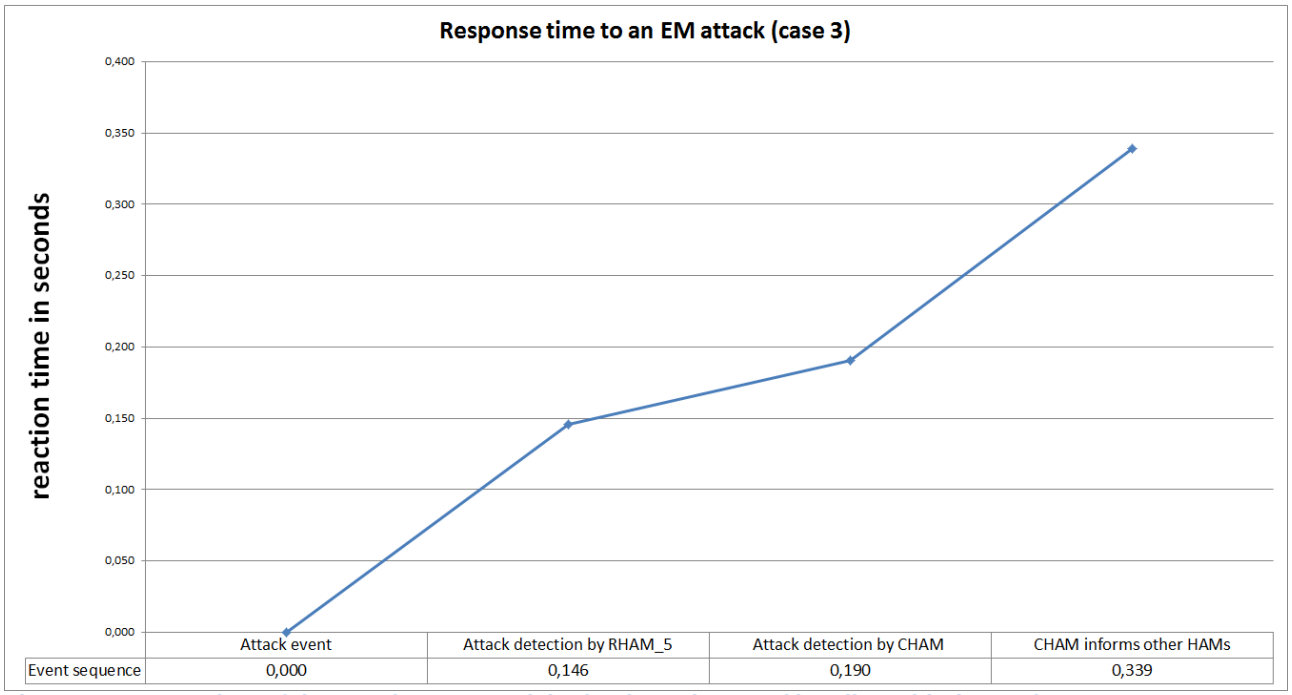


Fig. 61 Response time of the DPS in Base Model 5 (Train under attack) as line with data points

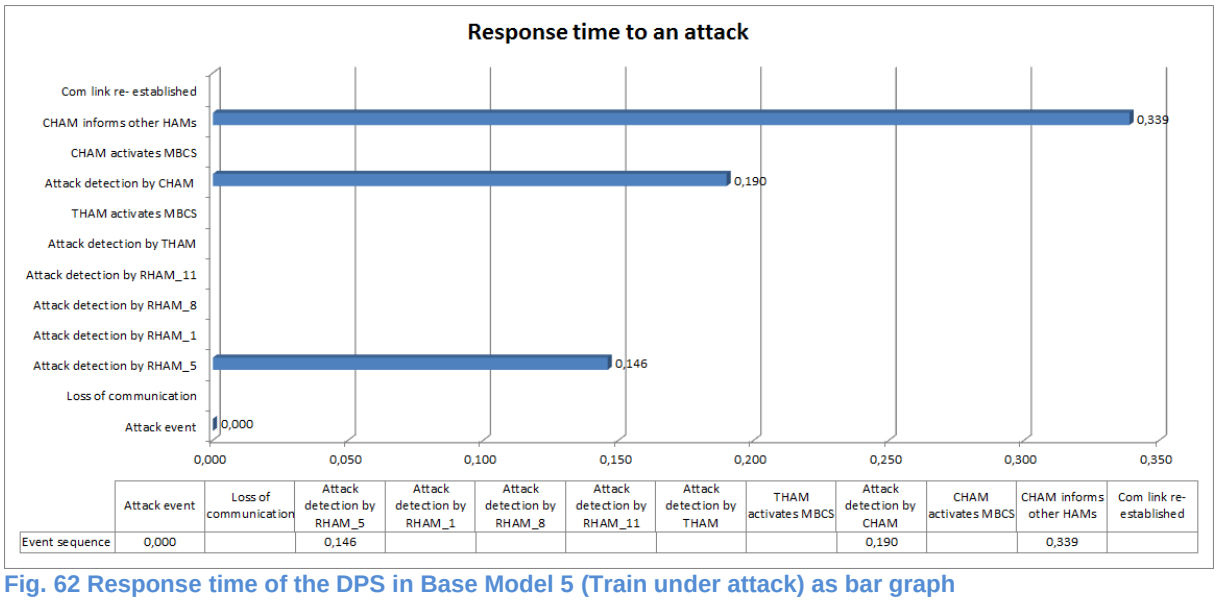


Fig. 62 Response time of the DPS in Base Model 5 (Train under attack) as bar graph

6.2.9SECRET Base Model_6

- Attacked device: RHAM_5 (track health/attack manager), THAM_0 (train)
- Attack severity: high Impact
- Attack duration: 10 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

In SECRET Base Model_6 we describe a completely new scenario where the attack occurs at several locations simultaneously. The attacked components are RHAM_5 at the track and additionally the train with an onboard jamming device.

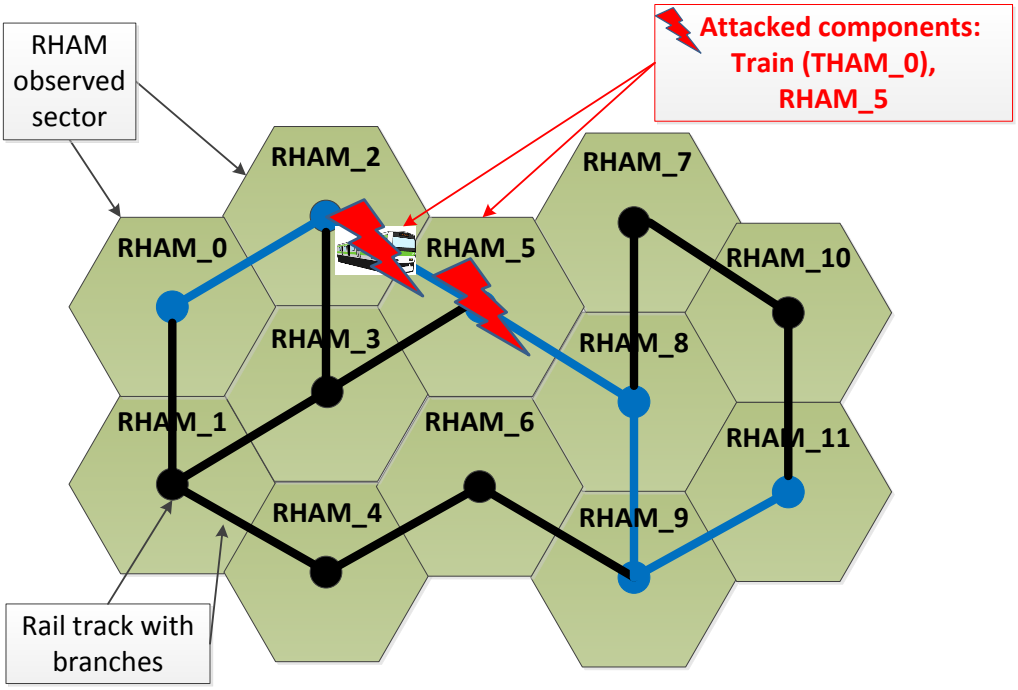


Fig. 63 Simulated attack scenario with RHAM_5 sector and train under attack

Result description

- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.064 seconds
- The communication connection between THAM and CHAM is lost 0.058 seconds after the attack event. The connection loss has no impact on the reaction time of CHAM because the KAT is already decreasing. In this case, only the KAT expiration triggers a CHAM reaction.
- It takes 0.207 seconds for the THAM to detect the EM attack. The THAM reacts directly on the connection loss. This results in an
  - *detection time = Com link loss (0.058s)+ Attack detection THAM (0.149s) = 0.207s*
- Because of the connection loss the CHAM cannot be informed by the THAM about the ongoing attack. But it gets the information from RHAM_5. So, the CHAM detects the EM attack after:
  - *RHAM detection time (0,163s) + CHAM detection time (0.045s) = 0.208s*

- The THAM activates its MBCS 0.149s after the attack has been detected by calculating
  - *Com link loss + Attack detection Time + THAM/MBCS activation time*
- The CHAM activates its MBCS 0.054 seconds after the attack has been detected by calculating
  - *Attack detection time + MBCS activation time = 0.262s*
- In parallel to that the CHAM informs other HAMs 0.163s after the attack has been detected by calculating
  - *Attack detection time + HAM information time = 0.371s*
- The reestablishment of communication is done after further 0.163s by calculating the specific corresponding times which result in the highest overall reaction time. Here the CHAM is very late with its attack detection moment. Communication is reestablished after
  - *MBCS activation time + Com re-establishment time = 0.511s*
- Conclusion: When the train is attacked and the communication is lost then it takes 0.511s until the communication between train and CHAM is reestablished.

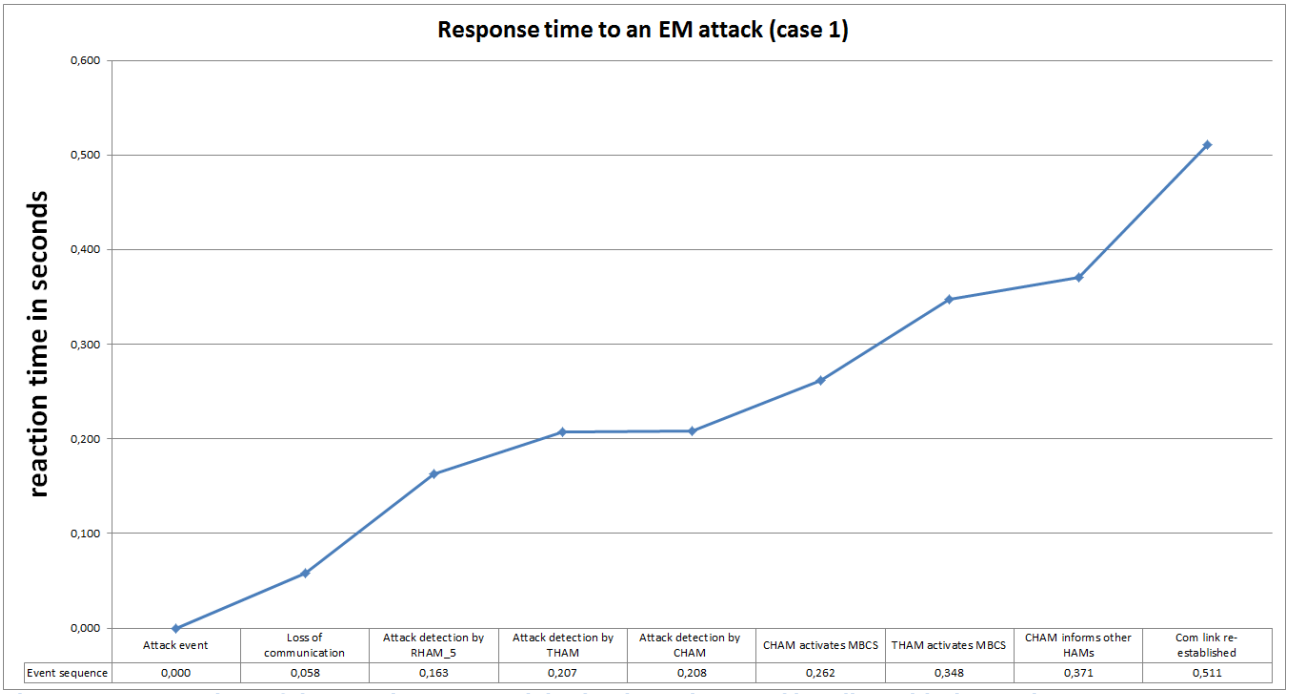
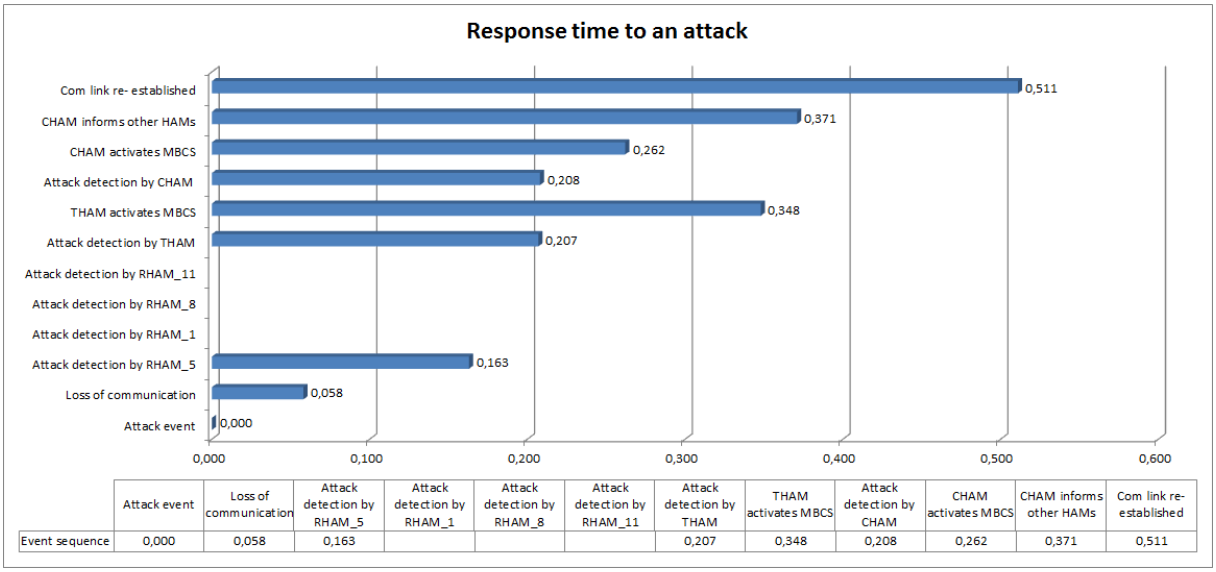


Fig. 64 Response time of the DPS in Base Model 6 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	THAM activates MBCS	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	Com link re-established
Event sequence	0,000	0,058	0,163				0,207	0,348	0,208	0,262	0,371	0,511

Fig. 65 Response time of the DPS in Base Model 6 (Train under attack) as bar graph

6.2.10 SECRET Base Model_7

- Attacked device: RHAM_5; RHAM_8 (track health/attack manager)
- Attack severity: high Impact
- Attack duration: 10 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

This scenario model describes an attack where not a train is involved but several railway health/attack managers. Here we want to show how the dynamic protection system behaviours when the attack is detected by a few neighbored RHAMs which are located on the train route.

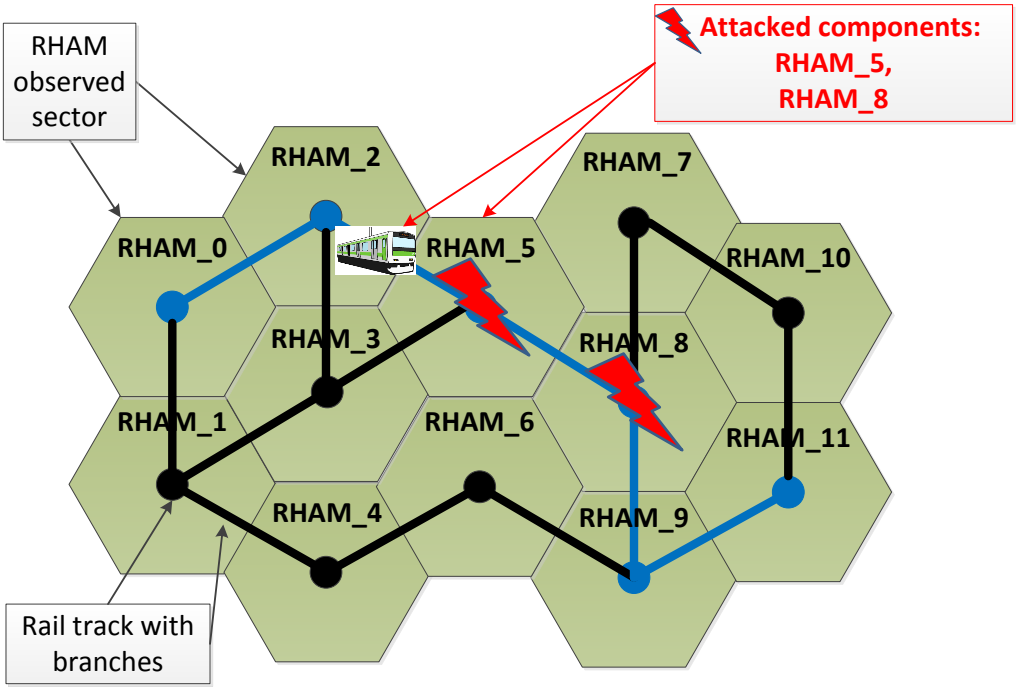


Fig. 66 Simulated attack scenario with RHAM_5 and RHAM_8 sector under attack

Result description
Case 1

- There is no keep alive timer expiration because the train is not driving in the attacked area of RHAM_5 or RHAM_8
- The communication connection between THAM and CHAM is not going to be broken
- It takes 0.149s for the RHAM_5 and 0.144s for the RHAM_8 to detect the attack
- The CHAM is informed about the ongoing attack 0.040s after the local RHAMs have detected it. The CHAM reacts on the first attack report sent by the earliest RHAM. So it completely takes for the CHAM
  - RHAM detection time (0.144s) + CHAM detection time (0.040s) = 0.184s
- Shortly (0.148s) after that the CHAM informs the HAMs in the railway topology about the attack. By this it takes 0.332s until all other HAMs have notice about the attack
- The CHAM activates its MBCS interface 0.057s later by calculating
  - CHAM detection time + MBCS activation time = 0.241s

- Because the THAM is in this case one the to be informed HAMs it needs at all 0.486s to activate its MBCS interface too
- And this is why the communication reestablishment takes completely 0.627s
- Conclusion: Here we can see that the attack report and MBCS activation process is initiated by a RHAM state change to CHAM. A keep alive timer is not running because the train is not directly affected. Because the train is in this scenario the last component in the information chain which receives the attack information via CHAM this component determines the delay until the communication is re-established. The CHAM is much earlier in a kind of waiting mode until the THAM accepts the new connection.

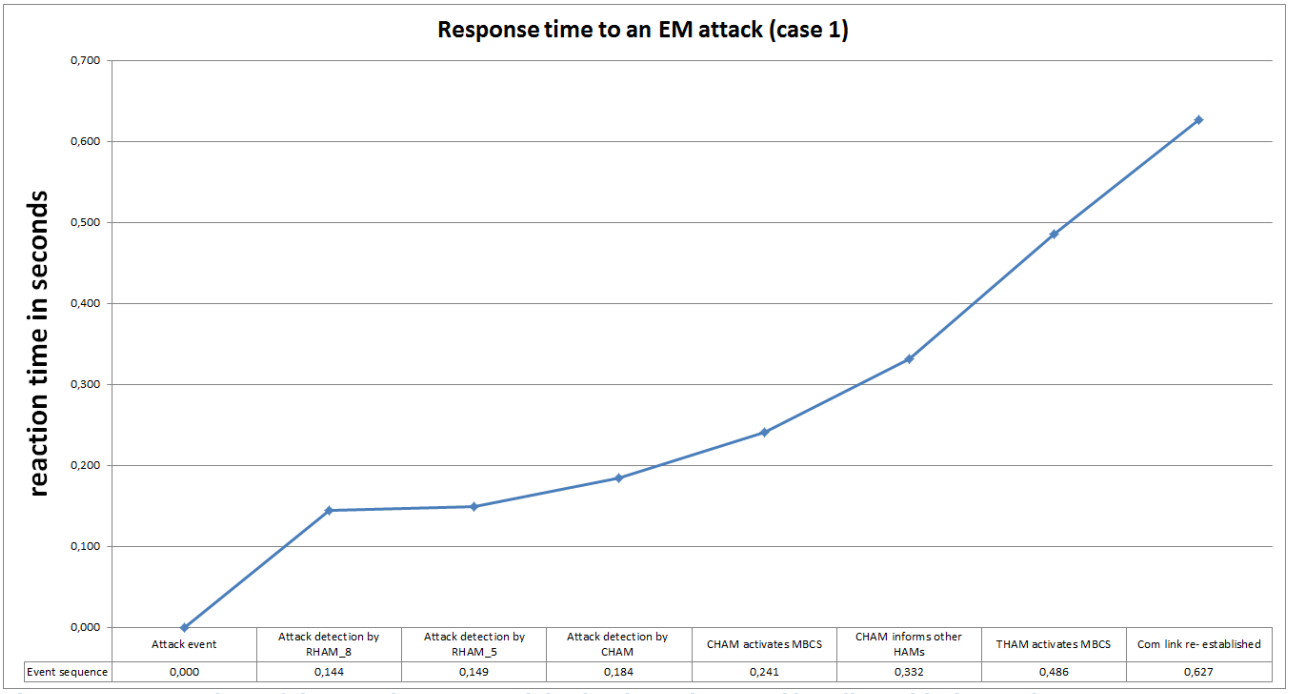


Fig. 67 Response time of the DPS in Base Model 7 (Train under attack) as line with data points

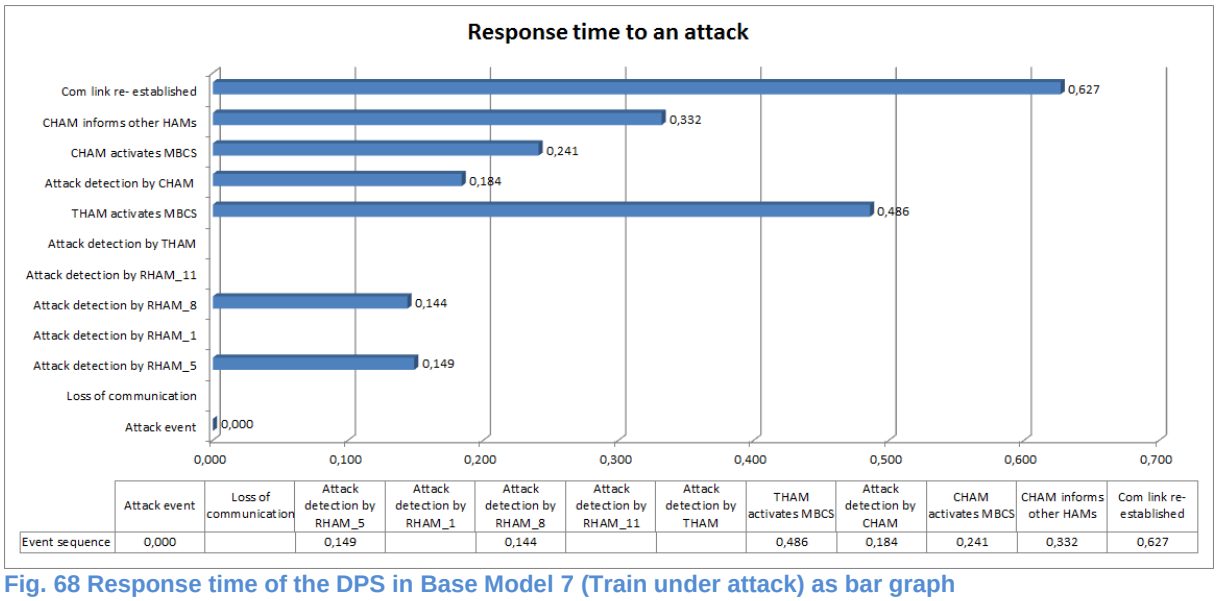


Fig. 68 Response time of the DPS in Base Model 7 (Train under attack) as bar graph

Case 2

- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.033 seconds
- The communication connection between THAM and CHAM is lost 0.094 seconds after the attack event. The connection loss has no impact on the reaction time of CHAM because the KAT is already decreasing. In this case, RHAM_5 and RHAM_8 will provoke the CHAM reaction
- RHAM_5 takes 0.149s and RHAM_8 takes 0.144s to detect the attack
- It takes 0.219 seconds for the THAM to detect the EM attack. The THAM reacts directly on the connection loss. This results in an
    - *detection time = Com link loss (0.094s)+ Attack detection THAM (0.125s) = 0.219s*
- Because of the connection loss the CHAM cannot be informed by the THAM about the ongoing attack. But it receives the attack information by one RHAM. So, the CHAM detects the EM attack after:
    - *RHAM detection time (0.144s) + CHAM detection time (0.0.040s) = 0.184s*
- The THAM activates its MBCS 0.154s after the attack has been detected by calculating
    - *Com link loss (0.094s) + Attack detection Time (0.125) + THAM/MBCS activation time (0.154s) = 0.373s*
- The CHAM activates its MBCS 0.057s seconds after the attack has been detected by calculating
    - *RHAM detection time + CHAM detection time + MBCS activation time = 0.241*
- In parallel to that the CHAM informs other HAMs 0.148s after the attack has been detected by calculating
    - *RHAM detection time + CHAM detection time + HAM information time = 0.332s*
- The reestablishment of communication is done after further 0.141s by calculating the specific corresponding times which result in the highest overall reaction time. Here the CHAM is very late with its attack detection moment. Communication is reestablished after
    - *MBCS activation time + Com re-establishment time = 0.514s*
- Conclusion: When the RHAMs are attacked and the train is at this moment driving within the scope of these RHAMs then the train will lose its communication connection to CHAM because the train is also affected by the attack. The train is no longer able to inform the CHAM about the attack. So the CHAM starts to monitor the train keep alive timer. This timer would expire after 5.033s. Because of the RHAM attack detection only 0.144s after the attack event the CHAM will be informed much earlier than waiting for the keep alive timer expiration. The CHAM immediately reacts on RHAM reports and starts to initialize the MBCS activation process. CHAM and THAM start the MBCS activation approximately at the same time. So it doesn't take very long to establish a new communication between them.

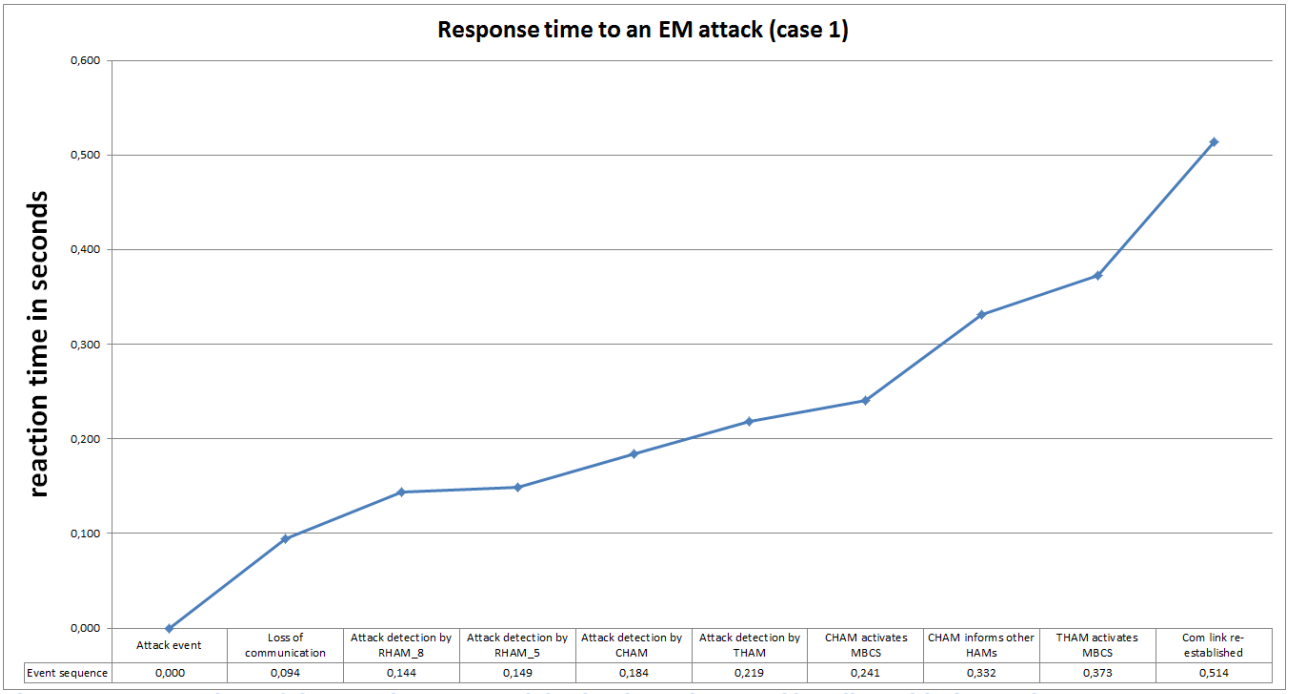


Fig. 69 Response time of the DPS in Base Model 7 (Train under attack) as line with data points

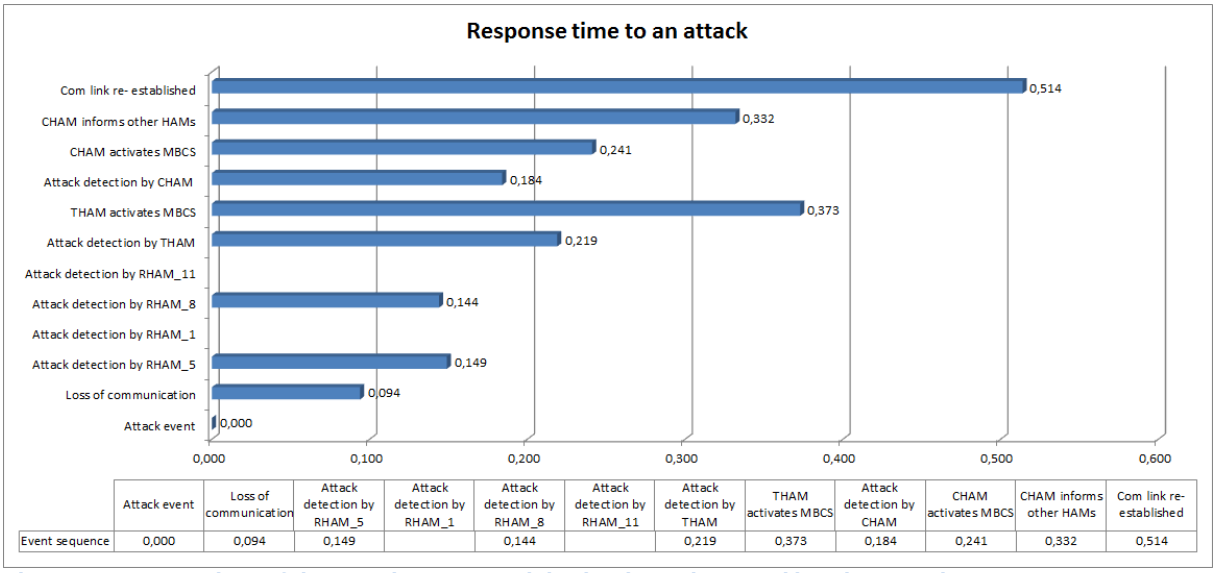


Fig. 70 Response time of the DPS in Base Model 7 (Train under attack) as bar graph

Case 3

- The train has already passed the attacked area of RHAM_5 and RHAM_8
- What we could measure is the attack detection time of RHAM_5 after 0.144s and the attack detection time of RHAM_8 after 0.149s. Depending on this follows the attack detection time of the CHAM after further 0.040s and the reporting process of the CHAM to inform all other HAMs in the railway topology.
- Conclusion: Because the train has already passed RHAM_5 and also RHAM_8 on its route there is no necessity for the train to initiate MBCS. The same thing

happens on the side of the CHAM. It knows that there is an attack in the topology but the train is not in danger for communication loss at the moment.

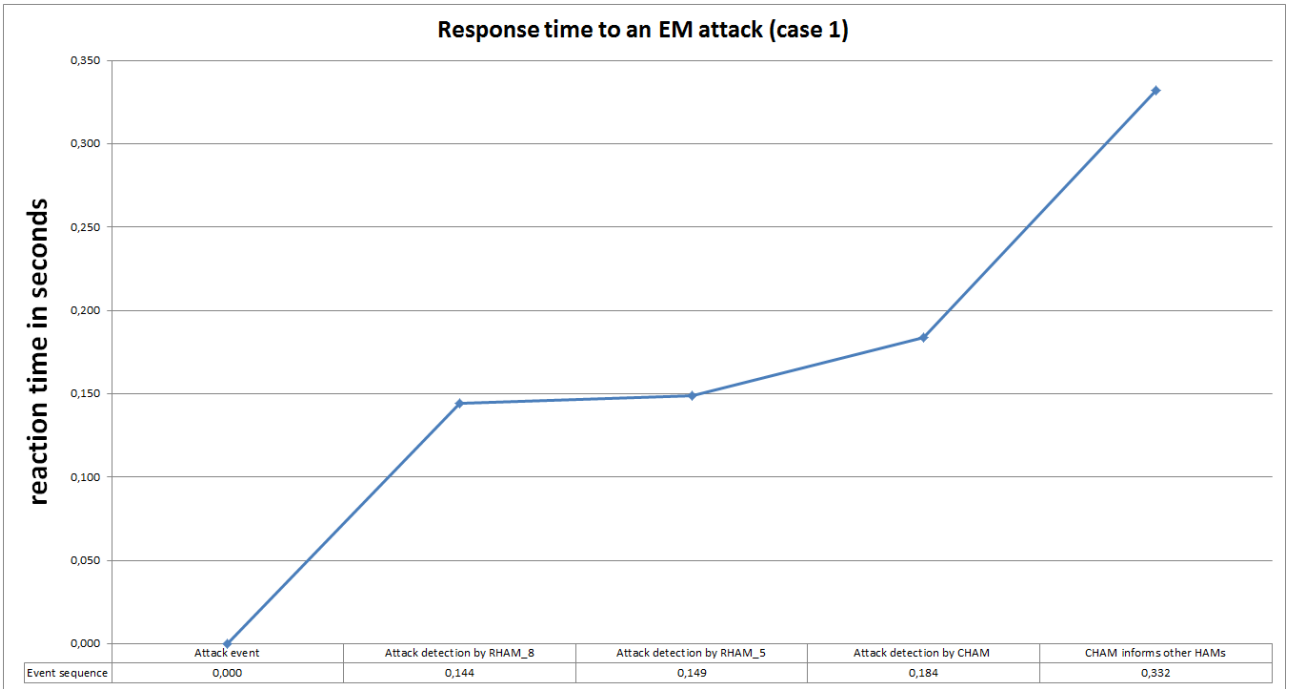


Fig. 71 Response time of the DPS in Base Model 7 (Train under attack) as line with data points

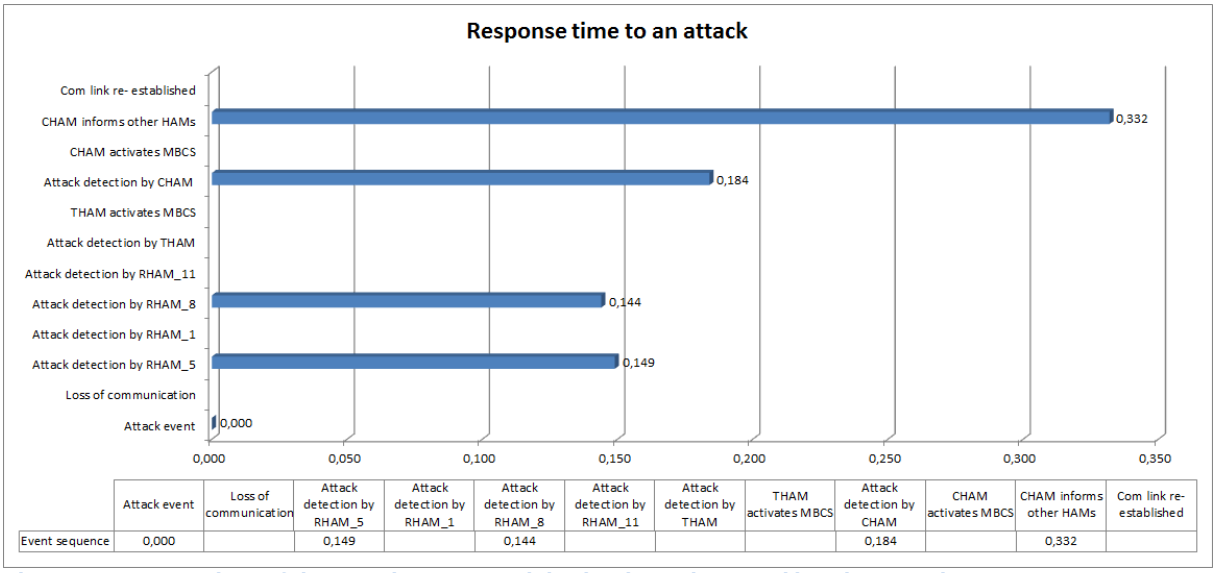


Fig. 72 Response time of the DPS in Base Model 7 (Train under attack) as bar graph

6.2.11 SECRET Base Model_9

- Attacked device: RHAM_5; RHAM_1; RHAM_11 (track health/attack manager), THAM_0 (train)
- Attack severity: high Impact
- Attack duration: 10 seconds
- THAM route: RHAM_0 -> RHAM_2 -> RHAM_5 -> RHAM_8 -> RHAM_9 -> RHAM_11

SECRET Base Model_9 is the most complex scenario model. We consider an EM attack event where a lot of RHAMs are involved. Some of them are located on the train route and another RHAM is beside the train route. Additionally the train is attacked, too. So we here have a comprehensive attack scenario where nearly the complete dynamic protection system is involved.

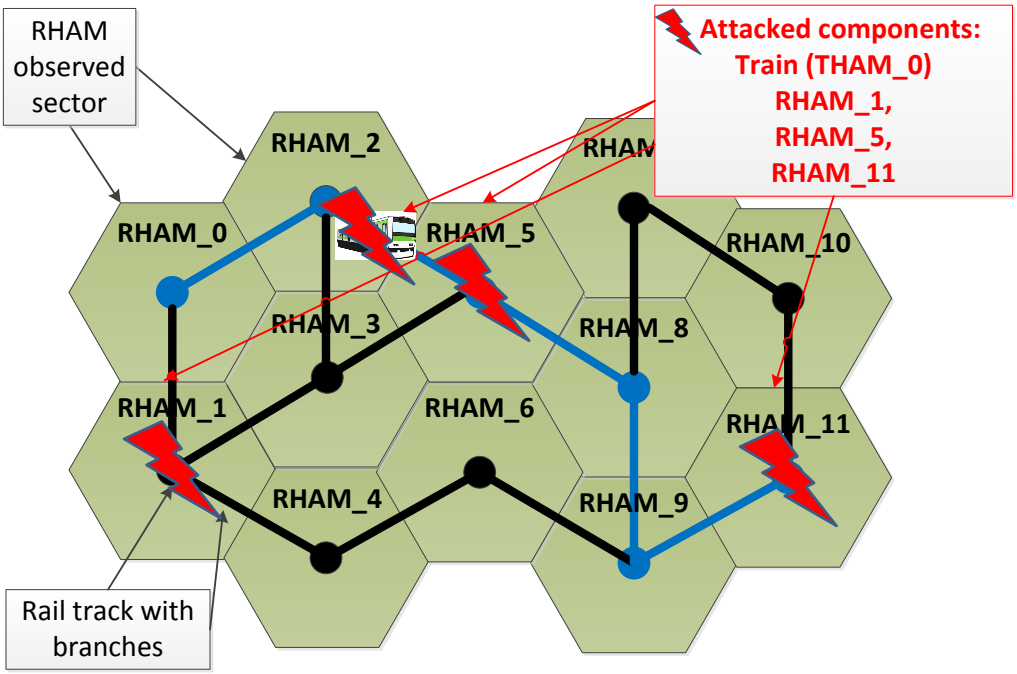
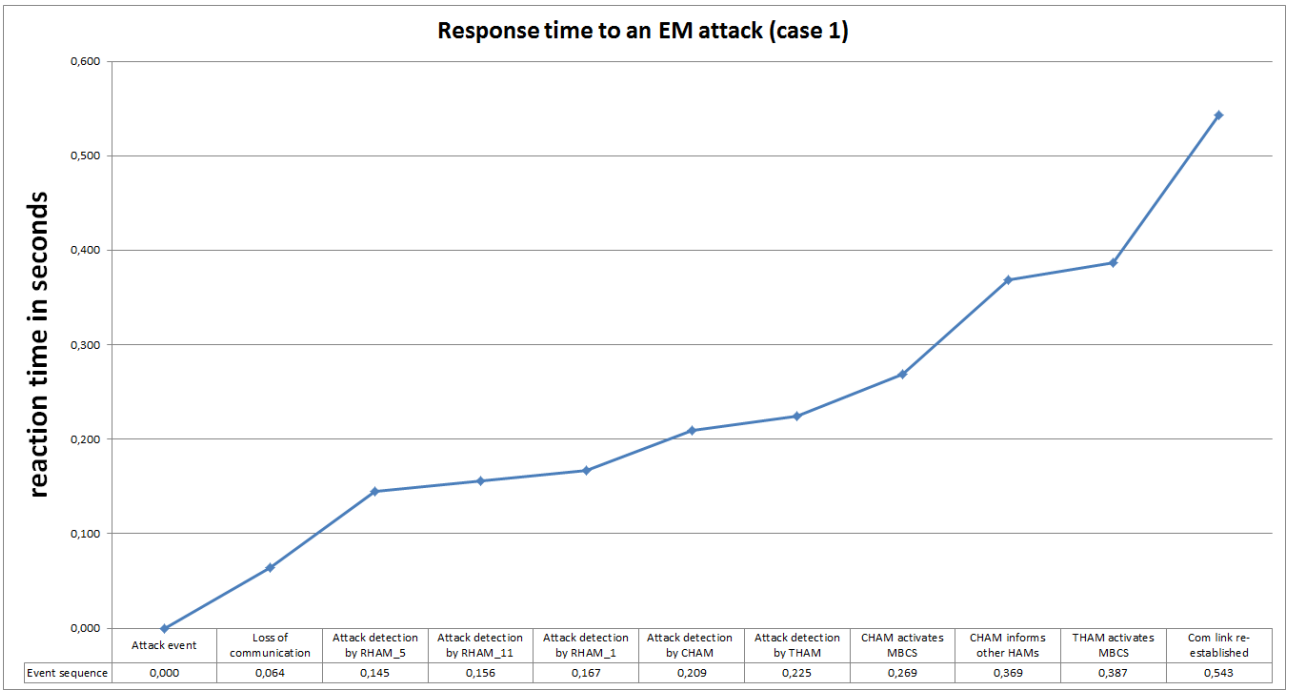


Fig. 73 Simulated attack scenario with train, RHAM_1/RHAM_5/RHAM_11 sector under attack

Result description

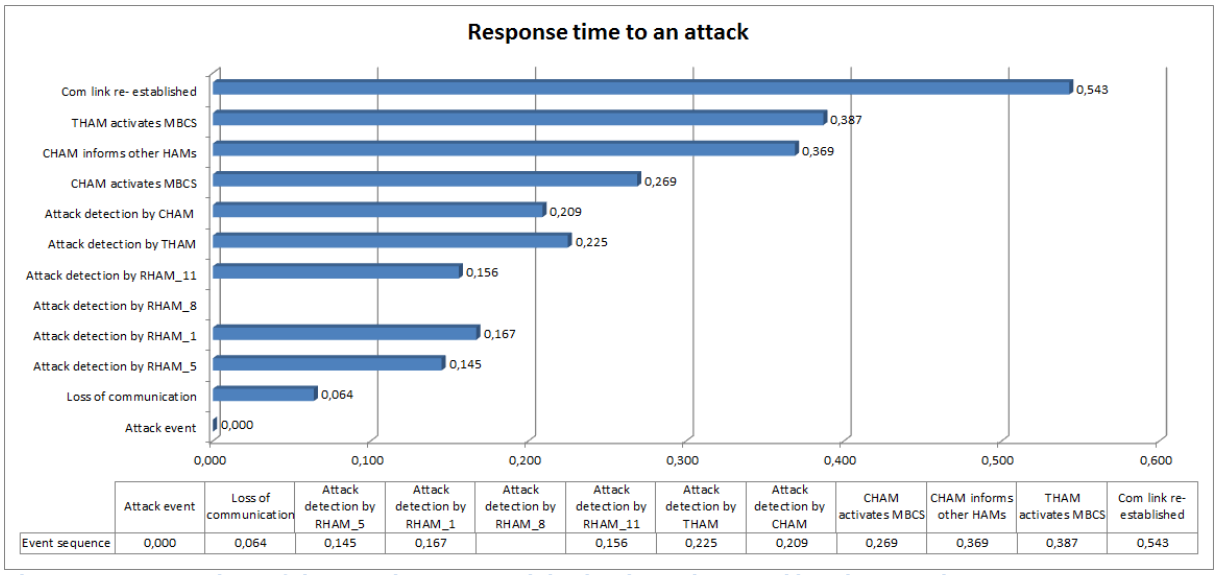
- In each attack process the keep alive timer (short: KAT) of the THAM which is monitored by CHAM expires after 5.073 seconds
- The communication connection between THAM and CHAM is lost 0.064 seconds after the attack event.
- Because the attack is performed at 4 attack locations simultaneously we have to consider 4 different attack detection times to evaluate the complete reaction time. RHAM_5 needs 0.145s, RHAM_11 needs 0.156s, RHAM_1 needs 0.167s and the train health/attack manager needs 0.225s to detect the attack. The train loses its connection and is not able to inform the CHAM
- Of course, the CHAM reacts to the earliest attack report. This results in a CHAM detection time by calculating one of the RHAM detection times
    - RHAM detection time (0.145) + CHAM detection time (0.064) = 0.209s
- CHAM activates the MBCS only 0.060s after the attack detection
- THAM also activates the MBCS after 0.162s
- The communication reestablishment is done after 0.156s. So it completely takes here
    - THAM MBCS activation time (0.387s) + reestablishment time (0.156s) = 0.543s
- Conclusion: The more RHAMs we simulate at the track the higher is the opportunity for the CHAM to detect an ongoing attack in a very short time after the attack event

occurred. The train always triggers the CHAM keep alive timer. This means when the train is also affected by the attack there is no chance to send an attack report from train to CHAM. The CHAM is reliant upon at least one RHAM report that also detects the ongoing attack which caused the train communication loss. In this case the CHAM is able to immediately activate its MBCS interface. Since the train has already switched the communication interface and the train is waiting for the CHAM MBCS activation the CHAM is here the delaying factor because it takes some more time until the attack report reaches the CHAM via different RHAMs. The overall communication reestablishment time stays in the frame of a few hundredth seconds, or concrete 0.543s.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_11	Attack detection by RHAM_1	Attack detection by CHAM	Attack detection by THAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re-established
Event sequence	0,000	0,064	0,145	0,156	0,167	0,209	0,225	0,269	0,369	0,387	0,543

Fig. 74 Response time of the DPS in Base Model 9 (Train under attack) as line with data points



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re-established
Event sequence	0,000	0,064	0,145	0,167		0,156	0,225	0,209	0,269	0,369	0,387	0,543

Fig. 75 Response time of the DPS in Base Model 9 (Train under attack) as bar graph

7 Conclusion

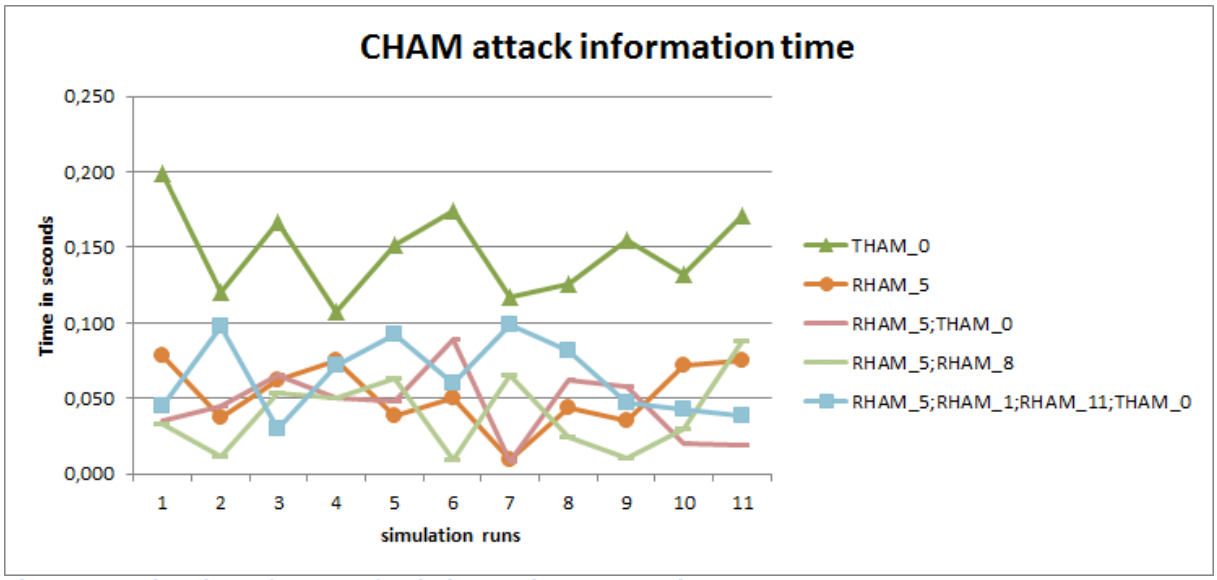


Fig. 76 Reaction time of CHAM after it detected an EM attack

Figure 76 shows the time when the CHAM is informed about the ongoing attack in different scenarios. With other words, the time stamp when the CHAM is in mode “EM attack is ongoing”. Here we don’t consider the keep alive timer (5s) which would distort especially the time line of THAM_0 (green line, THAM_0 is under attack). We also don’t consider the time how long it takes until a RHAM reports the attack information to the CHAM. The only thing you see here is the following:

- When a THAM is involved in the attack scenario and no other RHAM is attacked too, then the CHAM attack information time is approximately 100ms later than in the other cases when at least one RHAM is involved. This is caused by the communication architecture that we have implemented in our simulation. This implementation induces an additional state change within the CHAM agent. So the CHAM can set its state to “attack mode” in the following simulation cycle and not in the same cycle where it received the information.

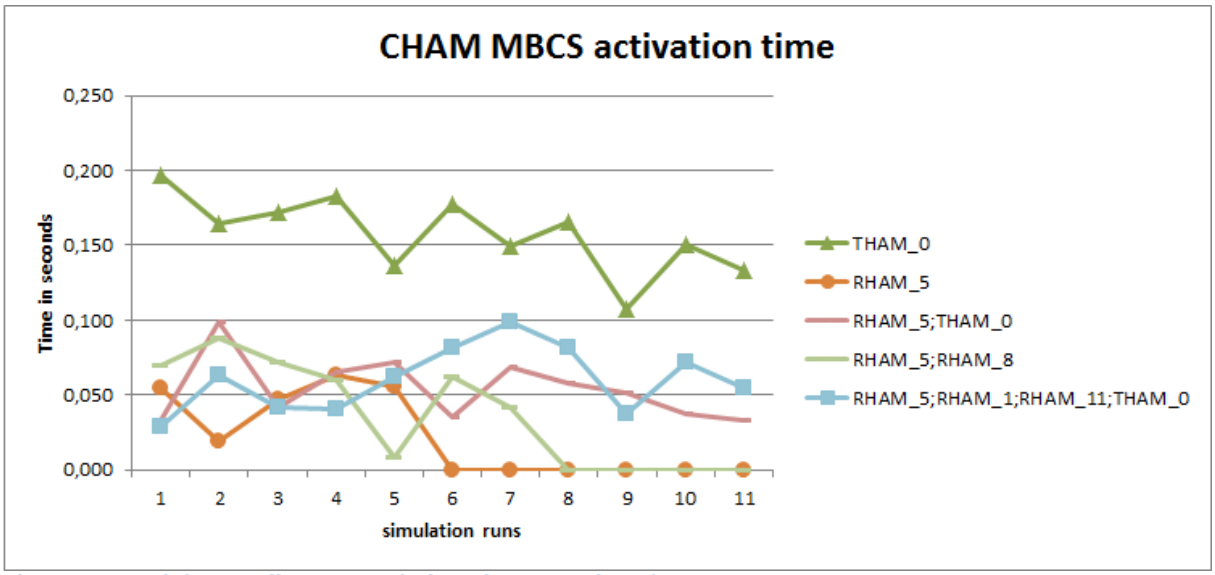


Fig. 77 Passed time until CHAM switches the MBCS interface

- The same behaviour occurs in the MBCS activation diagram (fig. 77) where the reaction time of the CHAM is delayed in a similar way.

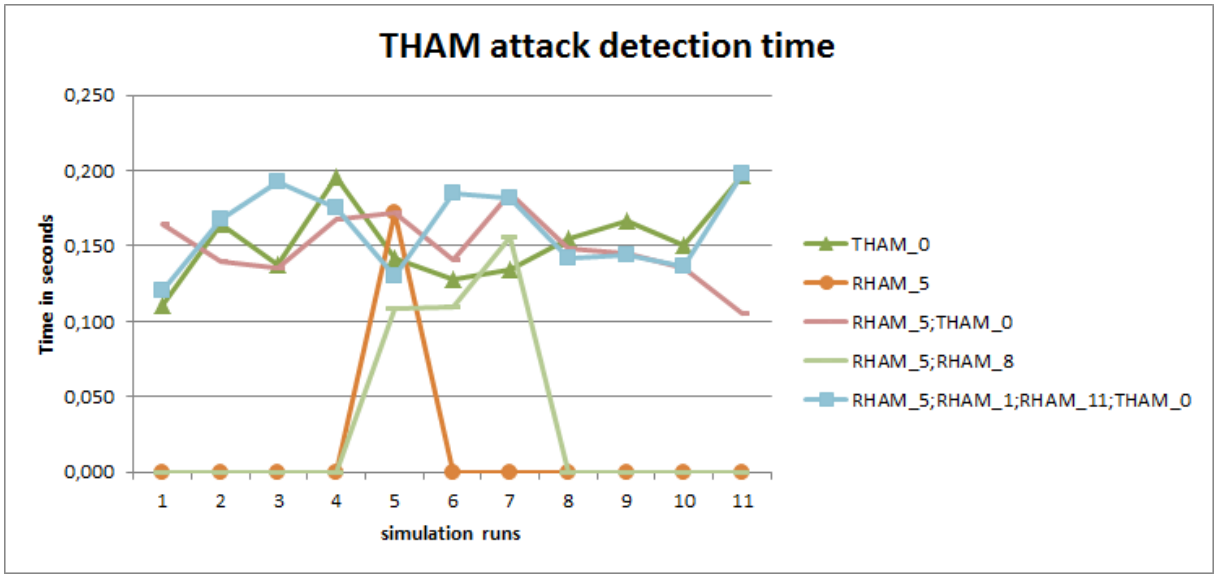


Fig. 78 Reaction time of THAM after it detected an EM attack

- The THAM attack detection time (fig. 78) proceeds in a time frame between 100ms and 200ms. Those diagram lines where THAM_0 is attacked directly are in this scope. In the lines where only RHAMs are attacked are a lot of zero-values because there is no attack detection time available for the train. The reason for this is that the train is not in the area of an attack at this moment. Only after the train enters an attacked area then it detects an ongoing attack and there is a reaction time.

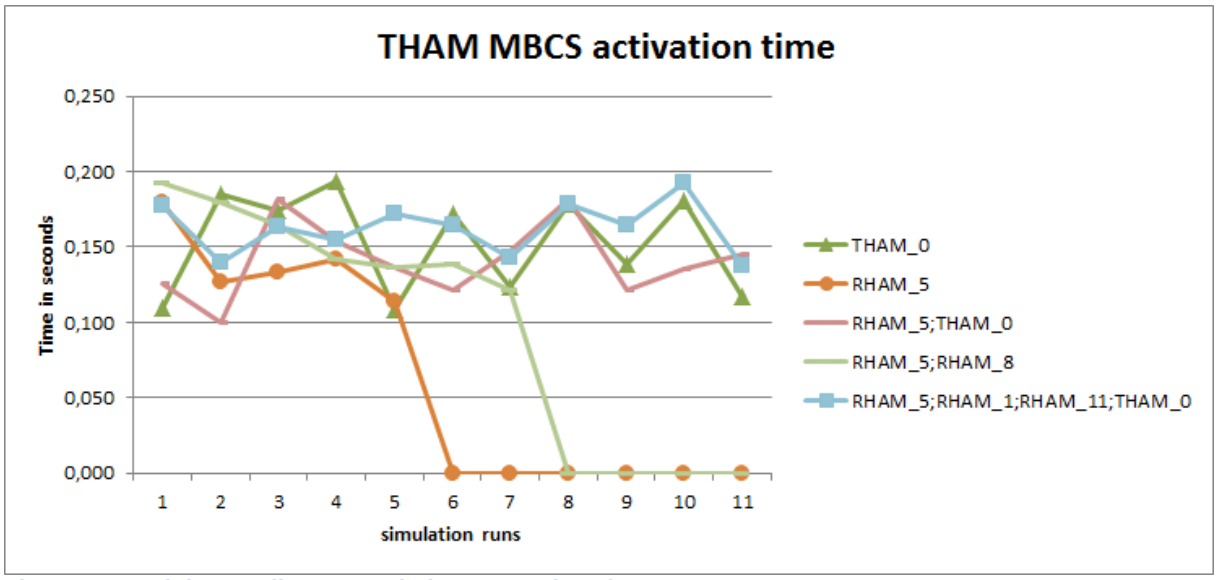


Fig. 79 Passed time until THAM switches MBCS interface

This diagram refers to the previous one and describes the train behaviour regarding the MBCS activation and interface switching process. We focus on the two lines (orange, light green) where the train is not directly attacked by a jammer. Although there is no attack detected by the train one can see a MBCS activation time (data column 1-4 in orange line; column 1-4 in light green line) which is in the same time frame as the other simulation scenarios. This results from the concrete CHAM behaviour. The CHAM receives the attack information from neighbored RHAMs. In a next step the CHAM informs the train about the ongoing attack. When the train analyses the attack information and recognizes that the attacked railway area is on the trains driving route then the train activates its MBCS before entering the attacked area. When the train has already passed the attacked area the train waives to switch the communication interface because it is not necessary.

Data column 5 of the orange line and data columns 5-7 of the light green line describe the situation where the train is inside the attacked RHAM area. This causes that the train detects the attack and also reacts in the same way by activating its MBCS.

In the following we give an overview description on how the Dynamic Protection System reacts in the different simulation scenarios.
We have considered various scenarios with several constellations of attacked devices. We simulated scenarios where an attack only happened onboard the train, only at one railway health/attack manager sector, at several railway health/attack manager sectors in parallel and several railway health/attack manager sectors plus train.
The overall conclusion that we found out is that the reaction time of the Dynamic Protection System depends very much on when the central health/attack manager receives the information about an ongoing attack. This time is very different in such scenarios where the train is directly targeted by an EM jamming device and those scenarios where the train is not directly under attack.

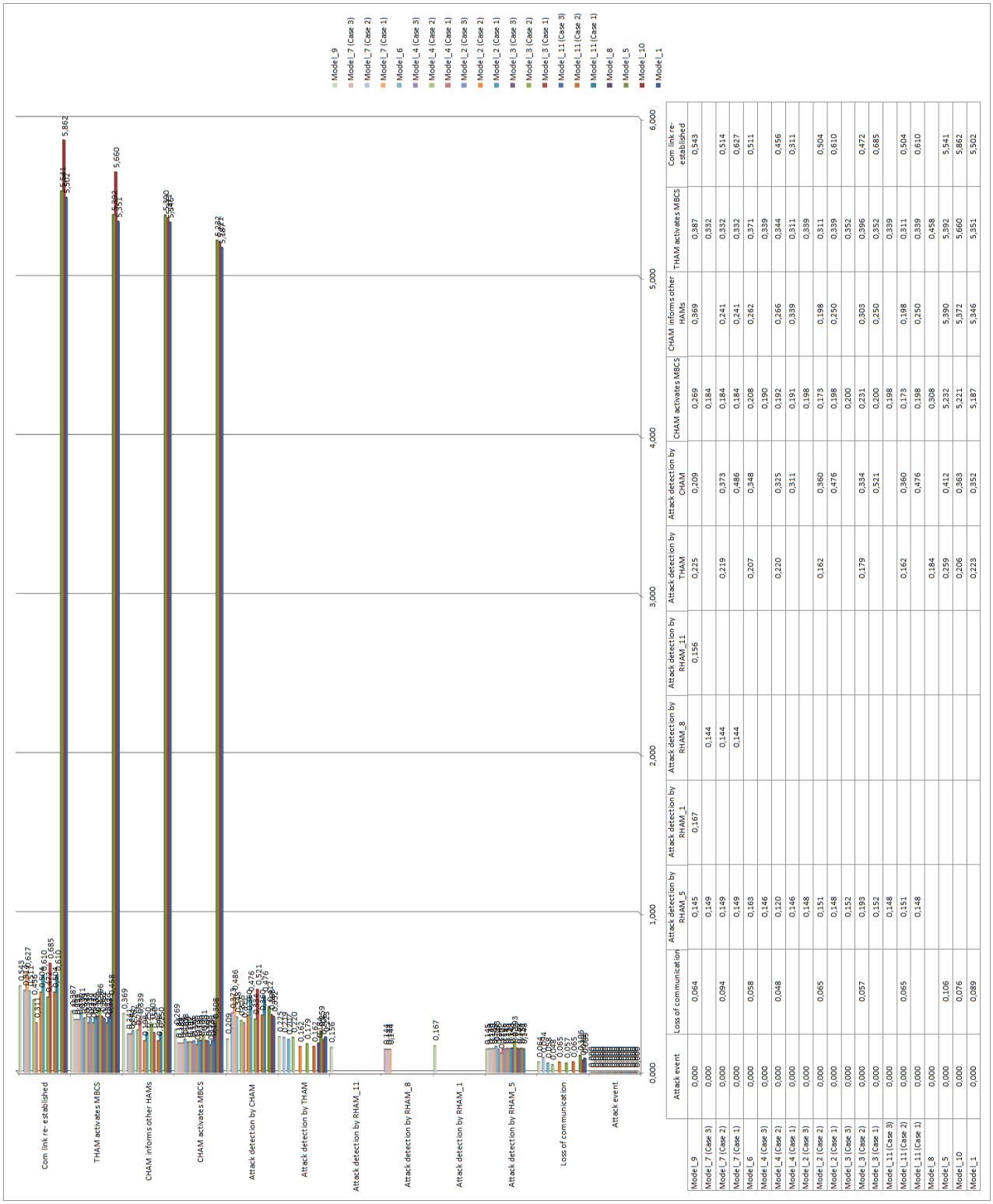
When only the train is in the scope of an attack (in case of an onboard jammer) this leads to the expiration of the train keep alive timer monitored by the CHAM. Depending on which time is set for the keep alive timer the CHAM delays its reaction according to that time. So, a keep alive timer of 5 seconds keeps the CHAM waiting for around 5 seconds, too. If we

decrease or increase the keep alive timer this has a direct impact on the CHAM reaction time and the reestablishment of a broken connection between train and CHAM.

When only a railway health/attack manager sector is going to be attacked by a jamming device the CHAM is approximately immediately informed about it via the *wired connection* between railway health/attack manager and CHAM. This leads to a fast reaction time of the CHAM. This case leads to a train information process because the CHAM has the task to inform the train about the ongoing attack. In our simulation this process takes one more simulation cycle. In reality there would also be a short delay until the train has notice about the attack. The result is that the train and CHAM can modify their communication interface much faster and establish a stable connection before the train enters the attacked area.

When the train and additionally one or more railway health/attack manager at the track are under attack we observe another behaviour. Of course, the train attack provokes keep alive timer expiration on the CHAM side. But here we can see that the railway health/attack manager detects the EM attack at the same time when the train detects it. The track detection system is wire connected to the CHAM. So there is no keep alive timer for the track detection system. This results in a much faster attack detection time by CHAM. The CHAM has notice of the EM attack before the train keep alive timer expires. It can immediately react on the attack event and initiate the MBCS interface switching process so that the communication between train and CHAM is reestablished in a shorter time.
As a matter of principle the CHAM responds to the attack information that reaches it first (first come, first serve). Attack reports that reach the CHAM later are ignored. This means, e.g., the keep alive timer anytime expires but the CHAM has already switched its MBCS communication interface triggered by a fast railway health/attack manager attack report.

Fig. 80 shows a complete view an all scenarios in the consideres SECRET Base Models. The individual diagrams from each scenario are added to this figure to show the temporal progress of the specific behaviours in comparison to all others.



	Attack event	Loss of communication	Attack detection by RHAM_5	Attack detection by RHAM_1	Attack detection by RHAM_8	Attack detection by RHAM_11	Attack detection by THAM	Attack detection by CHAM	CHAM activates MBCS	CHAM informs other HAMs	THAM activates MBCS	Com link re-established
Model_9	0,000	0,064	0,145	0,167		0,156	0,225	0,209	0,269	0,369	0,387	0,543
Model_7 (Case 3)	0,000		0,149		0,144				0,184		0,332	
Model_7 (Case 2)	0,000	0,094	0,149		0,144		0,219	0,373	0,184	0,241	0,332	0,514
Model_7 (Case 1)	0,000		0,149		0,144			0,486	0,184	0,241	0,332	0,627
Model_6	0,000	0,058	0,163				0,207	0,348	0,208	0,262	0,371	0,511
Model_4 (Case 3)	0,000		0,146						0,190		0,339	
Model_4 (Case 2)	0,000	0,048	0,120				0,220	0,325	0,192	0,266	0,344	0,456
Model_4 (Case 1)	0,000		0,146					0,311	0,191	0,339	0,311	0,311
Model_2 (Case 3)	0,000		0,148						0,198		0,339	
Model_2 (Case 2)	0,000	0,065	0,151				0,162	0,360	0,173	0,198	0,311	0,504
Model_2 (Case 1)	0,000		0,148					0,476	0,198	0,250	0,339	0,610
Model_3 (Case 3)	0,000		0,152						0,200		0,352	
Model_3 (Case 2)	0,000	0,057	0,193				0,179	0,334	0,231	0,303	0,396	0,472
Model_3 (Case 1)	0,000		0,152					0,521	0,200	0,250	0,352	0,685
Model_11 (Case 3)	0,000		0,148						0,198		0,339	
Model_11 (Case 2)	0,000	0,065	0,151				0,162	0,360	0,173	0,198	0,311	0,504
Model_11 (Case 1)	0,000		0,148					0,476	0,198	0,250	0,339	0,610
Model_8	0,000						0,184		0,208		0,458	
Model_5	0,000	0,106					0,259	0,412	5,232	5,390	5,392	5,541
Model_10	0,000	0,076					0,206	0,363	5,221	5,372	5,660	5,862
Model_1	0,000	0,089					0,223	0,352	5,187	5,346	5,351	5,502

Fig. 80 Block diagram with all Base Models sorted by the occurring behaviours.

References

[1] Rüdiger Klein, Uwe Beyer, Kai Pervölz, Alexander Zimmermann: "A Holistic Approach to Modeling, Simulation, and Analysis of Security in Modern Societies"; in: Proc. of the 8. Future Security Conference, Berlin, 2013.

[2] Klein, R., Rome, E., Beyel, C., Linnemann, R., Reinhardt, W. and Usov, A. (2008): 'Information modelling and simulation in large interdependent critical infrastructures in IRRIIS', 'n Proc. 3rd Int'l Workshop on Critical Information Infrastructures Security (CRITIS '08), Frascati, Italy, pp.41–62..

[3] Erich Rome, Sandro Bologna, Erol Gelenbe, Eric Luiijf, Vincenzo Masucci: DIESIS - Design of an Interoperable European Federated Simulation Network for Critical Infrastructures. In: Proceedings of the 2009 SISO European Simulation Interoperability Workshop (ESIW '09), Simulation Councils, Inc., San Diego, CA, USA, ISBN 1-56555-336-5, pp. 139-146.

[4] Rüdiger Klein, Jingquan Xie, Andrij Usov (2011): Complex events and actions to control cyber-physical systems. In: David M. Eyers, Opher Etzion, Avigdor Gal, Stanley B. Zdonik, Paul Vincent (Eds.): Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems, DEBS 2011, New York, NY, USA, July 11-15, 2011. ACM 2011 ISBN 978-1-4503-0423-8, pp. 29-38.

[5] Rüdiger Klein, Stefan Rilling, Andrij Usov, and Jingquan Xie: Using complex event processing for modelling and simulation of cyber-physical systems, in: Int. J. Critical Infrastructures, Vol. 9, Nos. 1/2, 2013.

[6] Michael Wooldridge: An Introduction to Multiagent Systems. John Wiley & Sons, 2009

[7] Gerhard Weiss (ed.): Multiagent Systems – a modern approach to Distributed Artificial Intelligence. The MIT Press, 2000

[8] Nigel Gilbert, Klaus Troitzsch: Simulation for the Social Scientist, McGraw-Hill International, 2005

[9] http://secret-project.eu/

[10] Klügl, F.; Oechslein, C.;Puppe, F.; Dornhaus, A.: Multi-Agent Modelling in Comparison to Standard Modelling. Simul. News Eur. 40, 3-9 (2004)

[11] E Jacob (EHU), M Higuero (EHU), C Pinedo (EHU), C Gransart (IFSTTAR), M Heddebaut (IFSTTAR), A Kung (TRIALOG), M Sall (TRIALOG): Preliminary specification of the dynamic protection system; D4.1, www.secret-project.eu (2013)

[12] E Jacob (EHU), M Higuero (EHU), C Pinedo (EHU), C Gransart (IFSTTAR), M Heddebaut (IFSTTAR), A Kung (TRIALOG), M Sall (TRIALOG): Final specification of the dynamic protection system; D4.2, www.secret-project.eu (2014)

[13] M. Heddebaut, J.P. Ghys, S. Ambellouis, D. Sodoyer, V. Deniau, S. Mili, V. Beauvois, H. Philippe: "Synthesis model" for experimental simulation of normal and attack conditions: methodology and results; D3.1; www.secret-project.eu (2013)