Security and Safety Modelling

Artemis JU Grant Agreement no.: 295354

D2.1 – Specification of Safety and Security Mechanisms
Version 0.1
29 May 2013
Final

Dissemination level: PU

Contributing partners
INT, IIT-CNR, AKH, DTU, FTW, ADEL, UC, CTU, PSA, SYSGO, IKV, ESY, IFAG, EADS, EDF, SAG, CITY

Project Partners:
Intecs, Institute of Informatics and Telematics - CNR, AKHELA, Università degli Studi di Roma La Sapienza, Technical University of Denmark, FTW Forschungszentrum Telekommunikation Wien, Adelard, UniControls, Czech Technical University in Prague, PSA Peugeot Citroën, SYSGO, ikv++ Technologies, eesy-id, Infineon Technologies AG Deutschland, EADS DEUTSCHLAND, Électricité de France, SYSGO s.r.o., Siemens AG Österreich, City University London - Centre for Software Reliability, General Motors Research & Development

Every effort has been made to ensure that all statements and information contained herein are accurate, however the Partners accept no liability for any error or omission in the same.
# Table of Contents

1 Introduction ........................................................................................................................................... 1
   1.1 Technical context and objectives ........................................................................................................... 1
   1.2 Document structure .................................................................................................................................. 2

2 Safety and Security Building Blocks .................................................................................................. 3
   2.1 Building block: Template.......................................................................................................................... 3
   2.2 Building block: Encryption/Decryption ..................................................................................................... 5
      2.2.1 Name.................................................................................................................................................... 5
      2.2.2 Description .......................................................................................................................................... 5
      2.2.3 Effects on safety .................................................................................................................................. 6
      2.2.4 Effects on security .............................................................................................................................. 6
      2.2.5 Trade-offs or synergies between safety and security ......................................................................... 6
      2.2.6 Interfaces (if applicable) ................................................................................................................... 6
   2.3 Building block: Signature Generation And Verification ................................................................. 7
      2.3.1 Name.................................................................................................................................................... 7
      2.3.2 Description .......................................................................................................................................... 7
      2.3.3 Effects on safety .................................................................................................................................. 8
      2.3.4 Effects on security .............................................................................................................................. 9
      2.3.5 Trade-offs or synergies for safety and security ................................................................................ 9
      2.3.6 Interfaces .......................................................................................................................................... 9
   2.4 Building block: Node Authentication ................................................................................................. 10
      2.4.1 Name.................................................................................................................................................... 10
      2.4.2 Description .......................................................................................................................................... 10
      2.4.3 Effects on safety .................................................................................................................................. 12
      2.4.4 Effects on security .............................................................................................................................. 13
      2.4.5 Trade-offs or synergies for safety and security ................................................................................ 13
      2.4.6 Interfaces .......................................................................................................................................... 13
   2.5 Building block: Access control/Traffic filtering ............................................................................. 14
      2.5.1 Name.................................................................................................................................................... 14
      2.5.2 Description .......................................................................................................................................... 14
      2.5.3 Effects on safety .................................................................................................................................. 15
      2.5.4 Effects on security .............................................................................................................................. 15
      2.5.5 Trade-offs or synergies for safety and security ................................................................................ 15
      2.5.6 Interfaces .......................................................................................................................................... 16
      2.5.7 Application notes .................................................................................................................................. 17
   2.6 Building block: Integrity protection ................................................................................................. 17
      2.6.1 Name.................................................................................................................................................... 17
      2.6.2 Description .......................................................................................................................................... 17
      2.6.3 Effects on safety .................................................................................................................................. 18
      2.6.4 Effects on security .............................................................................................................................. 18
      2.6.5 Trade-offs or synergies for safety and security ................................................................................ 18
      2.6.6 Interfaces .......................................................................................................................................... 19
      2.6.7 Application notes .................................................................................................................................. 19
   2.7 Building block: Checksums ............................................................................................................... 19
      2.7.1 Name.................................................................................................................................................... 19
      2.7.2 Description .......................................................................................................................................... 19
      2.7.3 Effects on safety .................................................................................................................................. 20
      2.7.4 Effects on security .............................................................................................................................. 20
      2.7.5 Trade-offs or synergies for safety and security ................................................................................ 20
      2.7.6 Interfaces .......................................................................................................................................... 21
<table>
<thead>
<tr>
<th>2.7.7</th>
<th>Application notes</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8</td>
<td>Building block: Bootchecks</td>
<td>21</td>
</tr>
<tr>
<td>2.8.1</td>
<td>Name</td>
<td>21</td>
</tr>
<tr>
<td>2.8.2</td>
<td>Description</td>
<td>21</td>
</tr>
<tr>
<td>2.8.3</td>
<td>Effects on safety</td>
<td>22</td>
</tr>
<tr>
<td>2.8.4</td>
<td>Effects on security</td>
<td>23</td>
</tr>
<tr>
<td>2.8.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>23</td>
</tr>
<tr>
<td>2.8.6</td>
<td>Interfaces</td>
<td>24</td>
</tr>
<tr>
<td>2.9</td>
<td>Building block: SW configuration checks</td>
<td>24</td>
</tr>
<tr>
<td>2.9.1</td>
<td>Name</td>
<td>24</td>
</tr>
<tr>
<td>2.9.2</td>
<td>Description</td>
<td>24</td>
</tr>
<tr>
<td>2.9.3</td>
<td>Effects on safety</td>
<td>25</td>
</tr>
<tr>
<td>2.9.4</td>
<td>Effects on security</td>
<td>25</td>
</tr>
<tr>
<td>2.9.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>26</td>
</tr>
<tr>
<td>2.9.6</td>
<td>Interfaces</td>
<td>26</td>
</tr>
<tr>
<td>2.10</td>
<td>Building block: Run-Time Monitoring</td>
<td>27</td>
</tr>
<tr>
<td>2.10.1</td>
<td>Name</td>
<td>27</td>
</tr>
<tr>
<td>2.10.2</td>
<td>Description</td>
<td>27</td>
</tr>
<tr>
<td>2.10.3</td>
<td>Effects on safety</td>
<td>28</td>
</tr>
<tr>
<td>2.10.4</td>
<td>Effects on security</td>
<td>28</td>
</tr>
<tr>
<td>2.10.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>29</td>
</tr>
<tr>
<td>2.10.6</td>
<td>Interfaces</td>
<td>29</td>
</tr>
<tr>
<td>2.10.7</td>
<td>Application notes</td>
<td>29</td>
</tr>
<tr>
<td>2.11</td>
<td>Building block: Plausibility Checks</td>
<td>30</td>
</tr>
<tr>
<td>2.11.1</td>
<td>Name</td>
<td>30</td>
</tr>
<tr>
<td>2.11.2</td>
<td>Description</td>
<td>30</td>
</tr>
<tr>
<td>2.11.3</td>
<td>Effects on safety</td>
<td>30</td>
</tr>
<tr>
<td>2.11.4</td>
<td>Effects on security</td>
<td>30</td>
</tr>
<tr>
<td>2.11.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>31</td>
</tr>
<tr>
<td>2.11.6</td>
<td>Interfaces</td>
<td>31</td>
</tr>
<tr>
<td>2.12</td>
<td>Building block: Logging</td>
<td>32</td>
</tr>
<tr>
<td>2.12.1</td>
<td>Name</td>
<td>32</td>
</tr>
<tr>
<td>2.12.2</td>
<td>Description</td>
<td>33</td>
</tr>
<tr>
<td>2.12.3</td>
<td>Effects on safety</td>
<td>33</td>
</tr>
<tr>
<td>2.12.4</td>
<td>Effects on security</td>
<td>33</td>
</tr>
<tr>
<td>2.12.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>33</td>
</tr>
<tr>
<td>2.12.6</td>
<td>Interfaces</td>
<td>34</td>
</tr>
<tr>
<td>2.13</td>
<td>Building Block Security Audit</td>
<td>34</td>
</tr>
<tr>
<td>2.13.1</td>
<td>Name</td>
<td>34</td>
</tr>
<tr>
<td>2.13.2</td>
<td>Description</td>
<td>34</td>
</tr>
<tr>
<td>2.13.3</td>
<td>Effects on safety</td>
<td>35</td>
</tr>
<tr>
<td>2.13.4</td>
<td>Effects on security</td>
<td>35</td>
</tr>
<tr>
<td>2.13.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>35</td>
</tr>
<tr>
<td>2.13.6</td>
<td>Interfaces</td>
<td>36</td>
</tr>
<tr>
<td>2.13.7</td>
<td>Application notes</td>
<td>36</td>
</tr>
<tr>
<td>2.14</td>
<td>Building block: Levels of Operation</td>
<td>36</td>
</tr>
<tr>
<td>2.14.1</td>
<td>Name</td>
<td>36</td>
</tr>
<tr>
<td>2.14.2</td>
<td>Description</td>
<td>36</td>
</tr>
<tr>
<td>2.14.3</td>
<td>Effects on safety</td>
<td>37</td>
</tr>
<tr>
<td>2.14.4</td>
<td>Effects on security</td>
<td>37</td>
</tr>
<tr>
<td>2.14.5</td>
<td>Trade-offs or synergies for safety and security</td>
<td>37</td>
</tr>
<tr>
<td>2.14.6</td>
<td>Interfaces</td>
<td>38</td>
</tr>
<tr>
<td>2.14.7</td>
<td>Application notes</td>
<td>38</td>
</tr>
</tbody>
</table>
2.15 Building block: Information Flow Control ................................................................. 39
  2.15.1 Name .................................................................................................................. 39
  2.15.2 Description ....................................................................................................... 39
  2.15.3 Effects on safety ............................................................................................... 39
  2.15.4 Effects on security ........................................................................................... 39
  2.15.5 Trade-offs or synergies for safety and security ............................................... 40
  2.15.6 Interfaces ......................................................................................................... 40
  2.15.7 Application notes ............................................................................................. 40
2.16 Building block: Partitioning .................................................................................... 40
  2.16.1 Name ................................................................................................................ 40
  2.16.2 Description ....................................................................................................... 40
  2.16.3 Effects on safety ............................................................................................... 41
  2.16.4 Effects on security ........................................................................................... 41
  2.16.5 Trade-offs or synergies for safety and security ............................................... 41
  2.16.6 Interfaces ......................................................................................................... 42
  2.16.7 Application notes ............................................................................................. 42
2.17 Building block: Virtualisation .................................................................................. 43
  2.17.1 Name ................................................................................................................ 43
  2.17.2 Description ....................................................................................................... 43
  2.17.3 Effects on safety ............................................................................................... 43
  2.17.4 Effects on security ........................................................................................... 43
  2.17.5 Trade-offs or synergies for safety and security ............................................... 43
  2.17.6 Interfaces ......................................................................................................... 44
2.18 Building block: Protocols for secure real-time communication .......................... 45
  2.18.1 Name ................................................................................................................ 45
  2.18.2 Description ....................................................................................................... 45
  2.18.3 Effects on safety ............................................................................................... 45
  2.18.4 Effects on security ........................................................................................... 45
  2.18.5 Trade-offs or synergies for safety and security ............................................... 45
  2.18.6 Interfaces ......................................................................................................... 46
2.19 Redundancy and diversity ....................................................................................... 46
  2.19.1 Name ................................................................................................................ 46
  2.19.2 Description ....................................................................................................... 46
  2.19.3 Effects on safety ............................................................................................... 48
  2.19.4 Effects on security ........................................................................................... 48
  2.19.5 Trade-offs or synergies between safety and security ....................................... 49
  2.19.6 Interfaces ......................................................................................................... 50
  2.19.7 Matching the kind of diversity to the threats ................................................... 50
2.20 Overview of building blocks .................................................................................... 52
3 Behavioral Models for selected building blocks ....................................................... 54
  3.1 Modelling of partitioning in IMACT .................................................................... 54
    3.1.1 IMACT Overview ............................................................................................ 54
    3.1.2 Main components in IMACT ......................................................................... 54
    3.1.3 Partitioning in IMACT ................................................................................... 55
    3.1.4 Building Block “Partitioning” to configure IPM partition .............................. 56
    3.1.5 Generation “IPM Partition” configuration for runtime system .................... 57
    3.1.6 Summary ....................................................................................................... 57
  3.2 Modelling of “Plausibility checks” building block ................................................. 58
    3.2.1 Introduction ..................................................................................................... 58
    3.2.2 The SysML language ..................................................................................... 59
    3.2.3 Generic model of plausibility checks ............................................................. 61
    3.2.4 Cross-influence modelling ............................................................................. 62
3.2.5 Parameters of the behavioural model ........................................................................................................ 62

3.3 Modelling of Real-time communication & Security building block .......................................................... 62
   3.3.1 Cross-influence modelling .......................................................................................................................... 63
   3.3.2 Parameters of the behavioural model ......................................................................................................... 63
   3.3.3 Example: Speech (VoIP) ............................................................................................................................ 64

3.4 Modelling of Access Controls/Traffic filtering building block ............................................................... 64
   3.4.1 Cross-influence modelling .......................................................................................................................... 65
   3.4.2 Parameters of the behavioural model ......................................................................................................... 66

3.5 Modelling of Monitoring building block ...................................................................................................... 66
   3.5.1 Cross-influence modelling .......................................................................................................................... 69
   3.5.2 Parameters of the behavioural model ......................................................................................................... 70

3.6 Modelling of redundancy and diversity ....................................................................................................... 71
   3.6.1 Modelling a redundant/diverse function/component from the system viewpoint .................................. 73
   3.6.2 Probabilistic modelling ............................................................................................................................ 74

4 Initial detailed analysis of selected building blocks ...................................................................................... 77
   4.1 Considered building blocks: Authentication and real-time communication .............................................. 77
      4.1.1 Analysis in Railway Use-case .................................................................................................................. 77
      4.1.2 Analysis in Smart Grid Use-case ........................................................................................................... 84
      4.1.3 Analysis in Automotive (CAN bus) ......................................................................................................... 87
      4.1.4 Analysis in Medical use case .................................................................................................................. 88
      4.1.5 Analysis in Industrial Drives use case ..................................................................................................... 91
      4.1.6 Summary of used models ...................................................................................................................... 95
      4.1.7 Outlook and Generalisation ................................................................................................................... 96
   4.2 Monitoring in embedded systems ............................................................................................................... 96
      4.2.1 Analysis in use-case avionics .................................................................................................................. 96
      4.2.2 Analysis in Medical Use case ................................................................................................................ 100
      4.2.3 Summary of the used models ................................................................................................................ 101
      4.2.4 Outlook and Potential for Generalization ............................................................................................... 102
   4.3 Partitioning .................................................................................................................................................. 102
      4.3.1 Analysis in Avionics Use-Case ................................................................................................................ 102
      4.3.2 Analysis in use-case avionics .................................................................................................................. 102
      4.3.3 Analysis in railway .................................................................................................................................. 107
   4.4 Redundancy and diversity ........................................................................................................................... 113
      4.4.1 Example (1-out-of-2 redundancy/diversity): ........................................................................................ 113
      4.4.2 Avionics use case – Secure Gateway ................................................................................................... 116
      4.4.3 Analysis in Railway Use-case ................................................................................................................ 117
      4.4.4 Analysis in Industrial Drives use case .................................................................................................. 120
      4.4.5 Preliminary Conclusions, Outlook and Generalisation Potential ....................................................... 124

5 Conclusions ..................................................................................................................................................... 125

6 References ....................................................................................................................................................... 126

7 Abbreviations .................................................................................................................................................. 130
LIST OF FIGURES

Figure 2-1: Principle of signature generation and verification ......................................................... 8
Figure 2-2: Relation of Digital Signatures to Hashes and MACs .......................................................... 8
Figure 2-3: Overview of Authentication Schemes [16] ....................................................................... 12
Figure 2-4: General model of a plausibility checking system ............................................................... 32
Figure 2-5: Interactions for the Logging building block ....................................................................... 34
Figure 2-6: Attributes of building blocks ............................................................................................. 52
Figure 3-1: IMA Architecture in IMACT ............................................................................................. 55
Figure 3-2: Relevant configuration elements for the IPM partition ...................................................... 56
Figure 3-3: Example of the IPM configuration with Partitioning on Application level (see ATA38) and Platform level (see ATA 42) ................................................................................. 56
Figure 3-4 Example of IPM configuration with Partitioning on Application level (see ATA38) and Platform level (see ATA 42) ................................................................. 56
Figure 3-5 PikeOS instance on top of Freescale P4080 ....................................................................... 57
Figure 3-6: Requirement for plausibility checks in ISO 26262 Part 6 (Software) ............................... 58
Figure 3-7: Requirement for plausibility checks for calibration data in ISO 26262 Part 6 (Software) ................................................................................................................................. 59
Figure 3-8: SysML Package Structure .............................................................................................. 60
Figure 3-9: SysML model of plausibility check as generic block ......................................................... 61
Figure 3-10: Model of "Traffic filtering" building block ..................................................................... 65
Figure 3-11: Example architecture of where a Monitor could be deployed ........................................... 67
Figure 3-12: Logical model of replicated (and possibly diverse) implementation of a function C with inputs x1,x2,..., replacing a non-replicated implementation with n replicas of it ......................................... 72
Figure 4-1: Generic ESSI use case ....................................................................................................... 77
Figure 4-2 ESSI function ...................................................................................................................... 78
Figure 4-3: ESSI component diagram ............................................................................................... 80
Figure 4-4: Principle of CBC-MAC .................................................................................................... 82
Figure 4-5: Abstract view on simulation scenario for communication network ........................................... 86
Figure 4-6: Industrial Drives Demo Installation [Source: ALTERA] .......................................................... 91
Figure 4-7: Authentication within the Industrial Drives Use Case ............................................................. 92
Figure 4-8: Aircraft Security Domain according to ARINC 811 ............................................................... 97
Figure 4-9 Integrated gateway software architecture based on Integrated Modular Avionics (IMA) and Multiple Independent Levels of Security (MILS) .......................................................... 98
Figure 4-10 Gateway modules .................................................................................................................. 98
Figure 4-11: Basic concepts ....................................................................................................................... 100
Figure 4-12: IMA / AFDX system view and communication ...................................................................... 103
Figure 4-13: Integrated Modular Avionics Configuration Tool IMACT ©SYSGO ..................................... 103
Figure 4-14: On-board aircraft network domains (© ARINC-811) ............................................................ 105
Figure 4-15: Integrated gateway software architecture based on IMA and MILS ..................................... 106
Figure 4-16: Generic use case of ESSI .................................................................................................... 108
Figure 4-17: Block diagram of the ESSI .................................................................................................. 109
Figure 4-18: Component diagram of ESSI (high level view) .................................................................. 110
Figure 4-19: PikeOS instance on P3041/2041 ......................................................................................... 112
Figure 4-20 Duplicated control system with different ............................................................................. 113
Figure 4-21: Generic use case of ESSI .................................................................................................. 118
Figure 4-22 Component diagram of ESSI (high level) ........................................................................... 119
Figure 4-23: Replication within the Industrial Drives Use Case ............................................................ 121

LIST OF TABLES
Table 2-1. Asymmetric and symmetric encryption algorithms and their comparable strength .......... 6
Table 2-2. Example of ACL model implementation as firewall rules ..................................................... 17
Table 2-3: Boot check - chain of trust ...................................................................................................... 22
Table 2-4: Boot check - Operating software validation ........................................................................... 22
Table 2-5: Safety and security key notions ............................................................................................... 38
Table 2-6 Overview of building blocks .......................... 53
Table 4-1: Security levels of authentication methods .................. 84
Table 4-2: Software components and security measures .................. 89
Executive Summary

This deliverable provides definition of SESAMO building blocks (BB) for safety and security modelling. A building block definition consists of description of the BB, BB main interfaces, BB contribution to safety and/or security, as well as cross-influence between safety and security within the BB. The deliverable also provides a first attempt to quantify safety, security and their cross-influence. Some of the BBs are further analysed within particular modelling environments and/or tool-chains. The last major part of the deliverable is analysis of BB within SESAMO use-cases. This analysis provides first feedback on modelling approaches, which will be elaborated in the next phase of the SESAMO project.
1 INTRODUCTION

1.1 TECHNICAL CONTEXT AND OBJECTIVES

WP2 Mechanisms for safety and security is dedicated to the identification and characterization of building blocks for ensuring safety and security, related to the types of needs expressed in the use case scenarios. The cross-influence of safety and security properties are modelled and analysed, and, based on the acquired understanding of the interrelationship between these properties, enhanced building blocks are specified that make it easier to balance security and safety requirements.

This document is the first deliverable of WP2; its purpose is to capture the building blocks identified analysing the use case scenarios and requirements developed in WP1, the modelling approaches developed for a selection of the identified building blocks and an initial analysis of the cross-influences between safety and security properties.

A critical analysis of the use cases was performed within WP2 in order to identify the most relevant building blocks related to safety and security and to their cross-influence. The result of this analysis produced a list of building blocks that are (i) directly derived from the project use cases presented in D1.2 and (ii) proposed by partners to enhance the use cases themselves w.r.t. safety and/or security.

This list of building blocks is not meant to be exhaustive to build any arbitrary embedded system but it is has been chosen to meet the SESAMO objectives and to achieve the SESAMO results. Nevertheless, the project team is confident that most of the interesting functionalities used to build embedded systems will be taken into consideration in WP2 and WP3 due to the large number of use cases in the project and to the fact that they are very cross-domain. This will eventually allow to build other systems using the project results.

Furthermore, restrictions have not been placed on the tools introduced in the context of individual building blocks. It is recognized that certain building blocks involve techniques and technologies that are supported by specialized tools and it is not realistic to artificially constrain their choice; rather, the challenge is elevated to a higher level within SESAMO, the level of the overall tool-chain approach addressed in WP4, to achieve an appropriate integration level of the tools (tight coupling, loose coupling, or some suitable combination).

The level of detail in the presentation and analysis varies according to the historical context associated with their use by partners responsible for their introduction:

- In some cases, considerable background work had been performed which was deemed relevant and appropriate to bring into the SESAMO context immediately;

- In other cases, regardless of background work, it was already clear how the building blocks could be applied to the individual use cases and it was considered useful to begin detailed analysis with respect to those individual use cases;

- In yet other cases, detailed background work was not available, or application to individual use cases was still under discussion, and therefore their description in this document is less detailed and specific to use case application.
This is appropriate and congruent with the purpose of the deliverable both to document the available consortium knowledge in terms of building blocks and to stimulate the discussion around their eventual application to the SESAMO use cases in subsequent project work.

1.2 DOCUMENT STRUCTURE

Chapter 2 is dedicated to building block descriptions:

- Section 2.1 introduces the definition of building block within the SESAMO context and from the SESAMO tool-chain point of view;
- Sections 2.2 to 2.18 present the selected building blocks. For each building block the followings are provided: a description, the effects on the safety and security of the system where it is used, a preliminary analysis of the trade-offs and synergies between safety and security, its interfaces to the environment and some application notes;
- Section 2.19 provides a first quantitative evaluation of safety and security cross influences in the different building blocks.

Chapter 3 deals with the behavioural modelling of selected building blocks. Each section provides the general model of a building block, some information about the cross-influence modelling and the main parameters of the behavioural model; an example of the modelling approach is also presented.

Chapter 4 presents an initial detailed analysis of some building blocks. Each building block is analysed from the perspective of the use cases exploiting it, identifying preliminary results and future actions.
2 SAFETY AND SECURITY BUILDING BLOCKS

In the WP2.1 we focused on defining “building block” in a pragmatic way to ease defining the scope and development of the architecture for the SESAMO tool-chain. A building block (BB) is a type of component, or architectural pattern, design solution, algorithm, protocol, or a category thereof, such that it can be analysed about at an abstract level (i.e., independent of a specific system). The analysis of the BB targets at defining functional characteristics as well as the identification of synergies and trade-offs between safety and security goals. The results of this analysis can then be plugged into the analysis or model of any specific system, e.g., as

- parameter values (e.g. failure probabilities, "integrity/assurance levels"),
- assumptions (e.g. "if you have ensured that predicates A, B are true for the building block, then you can assume that it deterministically ensures property X in the system"), or
- identification of what compromise level is best (along some dimension[s] where a trade-off is needed) in view of a certain objective function (e.g. total risk for system operation)

The pragmatic approach for WP2 is based on the following ideas:

- All BBs considered in the SESAMO project are derived from the SESAMO use-cases

  The selection of BBs identifies problems relevant to SEAMO project and be solved within the scope of the SESAMO project. Thus, we don’t target creation of a catalogue of all security controls or safety mechanisms (which can be found in standards and handbooks) nor a rigorous taxonomy of design features for achieving security or safety.

  However, our approach defines a methodology that can be applied across different systems, domains and use cases.

- The BBs scopes are defined with respect to the SESAMO use-cases.

  For example, the names “redundancy” and “runtime monitoring” can be seen as overlapping concepts but are treated in the WP2 as activities that solve specific analysis or design problems in SESAMO

- Analysis of a BB may ignore facts obvious to the SESAMO consortium or problems outside of the SESAMO technical or temporal scope.

We believe that our pragmatic approach will allow us to create a relevant list of BBs to support definition of the SESAMO tool-chain. At the same time, WP2 results can be used as methodology for analysis of safety and security in a structural way across use-case an domains.

2.1 BUILDING BLOCK: TEMPLATE

In this section we describe a template for a BB. The BB definition shall consist of

- **Name**: name of the BB

- **Description**: Description of the BB environment and functional description of the BB.
- **Effects on safety**: how usage of this BB in a system influence the system safety
- **Effects on security**: how usage of this BB in a system influence the system security
- **Trade-offs or synergies for safety and security**: preliminary analysis for trade-offs and synergies between safety and security.
- **Interfaces**: Interfaces to the environment, e.g. instantiation parameters of the BB, communication channels.
- **Application notes**: any issues which should be considered when using/analysing the BB
2.2 **BUILDING BLOCK: ENCRYPTION/DECRYPTION**

2.2.1 **Name**
Encryption/Decryption

In this BB we treat encryption and decryption as complementary parts of the same process.

2.2.2 **Description**
Encryption can be used to preserve confidentiality of information stored or being in transfer. The main difference to the access-control is that the attacker has access to the encrypted state of the information.

Encryption algorithms convert input data called plaintext into output data called ciphertext. Besides the length, output data do not yield any information about the plaintext. For every encryption algorithm there is also a decryption algorithm that transfers ciphertext back into plaintext [10].

There are two major approaches for encryption and decryption: symmetric and asymmetric approaches. In both approaches the encryption and decryption transformations are defined by keys. Symmetric encryption algorithms use the same keys (i.e. symmetric keys) for encryption and decryption. Asymmetric encryption algorithms use a public key for encryption and a private key for decryption. These two keys are distinct, i.e. these keys are asymmetric. Popular symmetric encryption primitives are the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES); the latter is considered to be secure only in its variant as Triple Data Encryption Algorithm (TDEA) using a 112 bit key.

Asymmetric encryption algorithms are usually used for signature generation and verification or the exchange of symmetric keys. Commonly used asymmetric encryption primitives include RSA and Elliptic Curve Cryptography (ECC) [11]. With a similar level of security ECC algorithms can use a shorter key and are more efficient. RSA keys are typically 1024 to 2048 bit long. A RSA key size of 1024 bits corresponds to an ECC key size of 160 bit.

More details on the strength of encryption and the key length can be found in the table below. An algorithm’s security is based on the best known attack on the cryptographic algorithm. The table below was taken from *NIST SP-800-57 Recommendation for Key management – Part 1* [13].

<table>
<thead>
<tr>
<th>Bits of security</th>
<th>Symmetric key algorithms</th>
<th>Asymmetric key algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DSA, D-H</td>
</tr>
<tr>
<td>80</td>
<td>2TDEA</td>
<td>L = 1024, N = 160</td>
</tr>
<tr>
<td>112</td>
<td>3TDEA</td>
<td>L = 2048, N = 224</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>L = 3072, N = 256</td>
</tr>
<tr>
<td>192</td>
<td>AES-192</td>
<td>L = 7680, N = 384</td>
</tr>
</tbody>
</table>
Table 2-1. Asymmetric and symmetric encryption algorithms and their comparable strength

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>AES-256</td>
<td>L = 15360, N = 512</td>
<td>k = 15360</td>
<td>f = 512 +</td>
</tr>
</tbody>
</table>

Security bits are estimating the computational steps required to break the algorithm or determine the keys. For example, an algorithm that provides 80-bits of security takes $2^{80-1} \times T$ of time on average to attack, e.g. to guess the key (T is amount required for one encryption of plaintext and comparison to the ciphertext). It is not recommended anymore to use algorithms with less than 80 bits of security.

### 2.2.3 Effects on safety

Encryption and decryption are complex processes that come with significant computational costs. In particular asymmetric encryption consumes substantial resources compared to symmetric algorithms with the same level of security. Thus, any kind of encryption causes delays that need to be considered in time critical safety applications. In many embedded systems safety-related messages have strict real-time requirements of only few milliseconds. Thus, any kind of encryption causes delays that needs to be considered in time critical safety applications and performance impacts for any kind of communication infrastructure should be analyzed.

### 2.2.4 Effects on security

Encryption is a major cryptographic mechanism for providing confidentiality as well as authenticity of information.

### 2.2.5 Trade-offs or synergies between safety and security

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Delay</td>
<td>Level of security</td>
</tr>
<tr>
<td></td>
<td>The computational effort for en- and decryption depends on the key length. According to [25] the processing time is increased by 14 % when the key length of Advanced Encryption Standard (AES) is increased from 128 to 192 bit.</td>
<td>The security level provided by an encryption algorithm depends e.g. on the length of the key. For a secure encryption algorithm a brute force attack on the key should be the attack that requires least effort. For a 128 bit key there are $2^{128}$ possible keys for a 192 bit key there would be even $2^{192}$ possibilities increasing the effort for a successful attack significantly.</td>
</tr>
<tr>
<td>Synergies</td>
<td>Cryptographic checksums for fault detection, e.g. CBC-MAC (see Section 2.6)</td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.6 Interfaces (if applicable)

Encryption
2.1 Specification of Safety and Security Mechanisms

- **Input:**
  - Plaintext
  - Symmetric Key or Public Key
- **Output:**
  - Ciphertext

Decryption

- **Input:**
  - Ciphertext
  - Symmetric Key or Private Key
- **Output:**
  - Plaintext

2.3 **BUILDING BLOCK: SIGNATURE GENERATION AND VERIFICATION**

2.3.1 **Name**
Signatures

2.3.2 **Description**
In many cases there is a need to ensure that received data comes from the right source and was not altered on its way to the target over different network layers. The main concern in this case is to guarantee integrity and authenticity of a message. In this case the data as such can be transmitted unencrypted if it is digitally signed. The most advanced methods for digital signature to verify the authenticity of a received data rely on the public key infrastructure PKI, which is a widely used asymmetric method. In digital signature, like the key agreement algorithm, a device uses a pair of keys, ‘sign private-key’ and ‘sign public-key’. Only the device knows its sign private-key whereas the sign public-key is distributed to all the communicating devices. A device signs the message using a signatures algorithm with its sign private-key to generate a signature. Usually this process is split in two steps. First a HASH is generated from the message and the signature is then generated for the HASH. Any device that has got the access to the sign public-key of the signed device can verify the authenticity and integrity of the data with the signature using the signature verification algorithm. If any third party modifies the data or signature, the verification fails. Since only the signed device knows its sign private-key, it will be impossible for any other device to forge the signature. The following Figure 2-1 illustrates the principle.
Examples of digital signature algorithms are RSA [14], [15], DSA [15] or ECDSA [15].

Digital signatures are strongly related to MAC where the major difference is that MAC uses symmetric cryptography. Sometimes the term signature, without the prefix digital, is used interchangeably to the term MAC within the literature. Obviously this can lead to confusion. Even Hashes, MACs and Digital Signatures are strongly related they address different cryptographic primitives, as shown in the following table.

<table>
<thead>
<tr>
<th>Cryptographic primitive</th>
<th>Hash</th>
<th>MAC</th>
<th>Digital Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Authentication</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-reputation</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Keys</td>
<td>none</td>
<td>symmetric</td>
<td>asymmetric</td>
</tr>
</tbody>
</table>

**Figure 2-1: Principle of signature generation and verification**

**Figure 2-2: Relation of Digital Signatures to Hashes and MACs**

### 2.3.3 Effects on safety

Advantages:

- Adding a digital signature to messages ensures that no unintended messages from invalid sources can cause malfunction of a component.

Disadvantages:

- If done only in SW, the calculation and verification of the signature adds additional delays. This can complicate the application in time critical cases and realtime systems.
• If additional HW is introduced, the HW failure modes of that component influence the safety of the system.
• Additional components for signature generation and check must fulfil certain SIL requirements since they can malfunction and decrease overall system safety.

* E.g. ISO 26262 states:

"This requirement applies to ASILs (A), (B), C, and D, in accordance with 4.3: The development of safety mechanisms that prevent dual point faults from being latent shall comply with:

a) ASIL B for technical safety requirements assigned ASIL D;

b) ASIL A for technical safety requirements assigned ASIL B and ASIL C; and

c) engineering judgement for technical safety requirements assigned ASIL A’’

### 2.3.4 Effects on security

**Advantages:**

• Since correctly signed messages always come from a known and trusted source this mechanism prevents an attacker from introducing harmful content.

**Disadvantages:**

• Since the messages are transmitted in plain format eavesdropping is possible to analyze the traffic in order to spy a system.

### 2.3.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>computation overhead increase system reaction time. E.g. breaking</td>
<td>For higher security level also higher computation efforts for the algorithm is needed.</td>
</tr>
<tr>
<td></td>
<td>distance in automotive.</td>
<td></td>
</tr>
<tr>
<td>Synergies</td>
<td>The signature algorithm ensures that the message content has not</td>
<td></td>
</tr>
<tr>
<td></td>
<td>been changed, no matter if this happened accidentally during</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transmission or as result of an attack. It has to be considered that</td>
<td></td>
</tr>
<tr>
<td></td>
<td>error correction is not possible with this approach.</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.6 Interfaces

**Sender:**

• **Input:**
  - private key
  - hashing algorithm
  - encryption algorithm
D2.1 Specification of Safety and Security Mechanisms

- message

  - Output:
    - digital signature

Receiver:

- Input:
  - public key
  - hashing algorithm
  - decryption algorithm
  - message with digital signature

- Output:
  - signature valid/invalid

2.4 BUILDING BLOCK: NODE AUTHENTICATION

2.4.1 Name
Node Authentication

2.4.2 Description
Authentication is a process that comes in several flavours, and so there are various methods known to provide authentication. There is entity authentication as well as message authentication. Thus a major difference between entity authentication and message authentication is that entity authentication involves two parties communicating actively. To provide this requirement an entity authentication scheme has to involve some kind of clock or timeliness. Authentication can be provided in an one-way, pair-wise, or broadcast fashion [16]. Figure 2-3 gives a very compact overview on available authentication approaches and compares them on following properties:

Secure relay channel:
A parallel secure communication channel is available to exchange a secret.

Key predistribution:
Both communication partners know their keys before the communication starts.

Trusted 3rd party:
A certification authority is responsible for key management. This protects against man-in-the-middle attacks.

Time Synchronization:
Here for the communication partners a service is available that can provide a common time base.

one-way/pair-wise:
This property differentiates if the sender and receiver authenticate each other or only the sender authenticates itself against the receiver.

broadcast:
Broadcast authentication is the process where one entity authenticates messages to several parties.
self enforce:
Protocols can be used in such a way if there is no active central guidance involved. However, it is up to the higher-level security protocols to implement self-enforcement
### Authentication

<table>
<thead>
<tr>
<th>Authentication Algorithm</th>
<th>Requirements</th>
<th>Authentication Method</th>
<th>Resource requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>secure relay channel</td>
<td>key pre-distribution</td>
<td>trust. 3rd party</td>
</tr>
<tr>
<td>Digital Signature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero Knowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Stamp (MAC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asymmetric (MAC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TESLA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamport</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guy Fawkes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote User</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 2-3: Overview of Authentication Schemes [16]

Within the context of SESAMO the special case of hardware node authentication is in focus, which is a form of entity authentication. This is based on scenarios where the trustworthiness of certain HW components within an embedded system must be guaranteed. For example the question how to ensure that only reliable sensors (and e.g. no faked data from inexistent sensors) are accepted within a sensor network is addressed within scope. The problem from the automotive world how to ban counterfeit components from car infrastructure is another representative.

#### 2.4.3 Effects on safety

Advantages:

- Authentication has positive impact on system safety since it prohibits that unauthorized components influence safety functionality either only at system startup or periodically during normal operation.
- Authentication is required for any form of replication as authentication ensures uniqueness of a component. Without authentication any replication renders useless as a component.
could forge multiple other components and influence results (and as mentioned above safety).

Disadvantages:

- The additional functions like key infrastructure and time synchronization need to fulfil the target SIL requirement.
- Additional computation efforts during authentication phase

### 2.4.4 Effects on security

**Advantages:**

- Authentication is one central foundation of security in any secure system. It permits attacks coming from unauthorized sources.

### 2.4.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
</table>
| **Trade-offs** | Computation Overhead reduces reaction time. E.g. breaking distance in automotive. | For higher security level also higher computation effort for the algorithm is needed which can have negative impact on real time behaviour. It is to distinguish between
- Resource problems (e.g. memory, time, ...) which can be compensated and
- Principle real time problems so security code is destroying determinism (e.g. no WCET can be determined any more in principle)
Here especially for Node Authentication code. |
| **Synergies** | Authentication mechanism reduces the risk that messages are unintentionally misrouted and processed by a wrong target. A little bit about system availability under attacks of the system would be helpful. | Authentication mechanism reduces the risk that attacker from outside can manipulate the system. |

### 2.4.6 Interfaces

The shown authentication methods are very different with respect to the requirements. Consequently a generic building block holds superset of all possible inputs which are not needed in all cases.
• **Input:**
  - authentication algorithm
  - key infrastructure and trusted 3rd party as key provider
  - time synchronization mechanism for advanced methods like timestamp MAC

• **Output:**
  - Indication if communication partner is trustworthy

## 2.5 BUILDING BLOCK: ACCESS CONTROL/Traffic filtering

### 2.5.1 Name
Access control/Traffic filtering

### 2.5.2 Description
Access control is the process of determining whether the request to access resources or data of the system should be granted or denied. More specific, access control is a way to associate a set of objects (e.g. resources – information, process) $O$, a set of subjects $S$ (e.g. any entity – a person, process or device), and a set of rights $R$ in the way that a rule $r \in R$ is enforcing how should a subject $s \in S$ interact with an object $o \in O$. An important requirement for access controls is ensuring protection against unauthorized access (disclosure) or modifications (and destruction) of systems resources and data, while at the same time making the system available to legitimate users.

Concept of access controls is closely related with identification and authentication (I&A) and authorization processes. I&A process verify that an identity is bound to the subject that makes a claim of identity and this process is always executed before authorization. Authorization allows us to specify to which objects the subject should be allowed or denied, and access controls enables managing of the access on different levels of granularity.

Furthermore, an access control (AC) system can be described on three different levels of abstraction: policies, models and mechanisms [1]. AC security policies specify high-level requirements such as how access is managed and who may access to the resource or data, i.e. policy is set of rules that say what is allowed and what not. The AC polices are enforced through AC security mechanism and formally presented using AC security models. AC models bridge the gap in abstraction between AC policy and AC mechanism, some of the most popular models and concepts are:

- **Role-based Access Control** [2] – in this model subjects are associated with a role, which has permissions to access to objects. Moreover, the roles can be organized in a hierarchy, a subject can have multiple roles and a role can have a multiple subjects and permissions.
- **Mandatory Access Control** – assigns a security label to each subject and object. A subject’s security label defines the security clearance, while object’s security label defines category of object (security classification). The subject is permitted to access to the object with the corresponding label. Models representing this category are Bell-LaPadula and Biba model.
- **Discretionary Access Control** – uses the identity of the subject (or subject´s properties) to decide whether to grant an access request. Furthermore, the object’s owner
defines which subject can access to the object, i.e. the subject has the ability of passing on their rights to other subjects (granting and revoking rights). The model describing this policy is called Access Control Matrix.

### 2.5.3 Effects on safety

Poorly defined access controls could cause unintentional/accidental file modifications or actions by legitimate user that could lead to safety incidents. Furthermore, centralized administration of access controls (e.g. RADIUS server) could be potential single-point of failure that would affect availability of the whole system as well as safety-critical operations. Failure dependency in access control models has been studied in [4].

### 2.5.4 Effects on security

In general, access control building block enhances security of a certain system by protecting against malicious access to a system and modification of data.

Different access control techniques applied on different level of a system will provide varying levels of security for the whole system and for a particular component in general. Also, important aspect of security in access controls is strength of authentication type, i.e. credentials (password, PIN, smart card, etc.) as well as security of database (or file system) where principal authentication credentials are stored. Often times an attacker is bypassing the access controls by attacking the credential storage or exploiting vulnerabilities of poor access controls implementation [4] (improper input validation, e.g. SQL injection attacks, termination of remote access sessions after certain time, improper AC configuration etc.).

### 2.5.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Trade-offs</th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability of safety operation</strong></td>
<td>- access control lockout mechanisms could prevent legitimate user to access the system in order to perform some safety related operations</td>
<td><strong>Access control protection mechanism</strong></td>
</tr>
<tr>
<td></td>
<td>- in some cases to stringent</td>
<td>- brute-force attack on access control could trigger lockout mechanism that will prevent an attacker to guess the password</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- extending safety-related system with access controls</td>
</tr>
</tbody>
</table>
access controls rules can prevent legitimate user to access the system
could cause problems by consuming additional resources (memory and CPU time). Which would in turn result in destroying determinism needed by safety applications (e.g. worst-case execution time)

<table>
<thead>
<tr>
<th>Synergies</th>
<th>Access controls are preventing potential attackers to enter the system. Furthermore, by imposing the set of rules they are restricting the legitimate users from accidental actions (file modifications, deletion, etc.)</th>
</tr>
</thead>
</table>

### 2.5.6 Interfaces

- **Definition of AC model**
  - List of all subjects - $S$
  - List of all objects - $O$
  - List of all rules – $R$

- **Security level of access controls**
  - Credential types and management – depending on the operational context of a subject and sensitivity of an object, different assurance level in authorization is required. Also, more critical operations require high level of assurance in the subject’s identity, i.e. strong credentials.
  - Risk of authorization implementation vulnerabilities (input attacks, session termination attacks, etc.)
  - Threats to access control [5] – masquerading by an entity appearing to be a proper subject, bypassing of access controls, and attacks on communications related parts of access controls, etc.

- **Impact of AC on safety properties**
  - Probability of AC failure and impact on the system
  - Model checking for proving the least privilege design principle (every subject of the AC should operate using the least set of privileges needed to complete the task) - minimizing the risk from accidental actions on safety critical objects
  - Impact of DoS attacks on AC availability and implications on safety
2.5.7 Application notes
Traffic filtering in general and packet filtering in particular represents one of the most common types of access controls. Firewalls that are placed in network devices perform packet filtering on different communication layers, e.g. MAC, IP, transport and application layers and their properties can be utilized for defining the packet filtering rules either for incoming or outgoing traffic. Firewall rules are usually represented by simple AC model called access control list (ACLs). An ACL model allows groups of objects or groups of subjects to be controlled together. Subjects and objects in firewall rules usually present group or range of IP addresses.

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Destination</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0.12.0/24</td>
<td>192.168.10.0/24</td>
<td>DENY</td>
</tr>
<tr>
<td>2</td>
<td>10.0.12.0/24</td>
<td>192.168.10.10</td>
<td>ALLOW</td>
</tr>
</tbody>
</table>

Table 2-2. Example of ACL model implementation as firewall rules

2.6 BUILDING BLOCK: INTEGRITY PROTECTION

2.6.1 Name
Integrity protection

2.6.2 Description
The aim of integrity protection mechanisms is to protect data from unauthorized modifications including creation and deletion. In general data integrity can be threatened in the following ways [6]:

- Unauthorized data modification
- Unauthorized data deletion
- Unauthorized data creation
- Unauthorized data insertion
- Unauthorized data replay

Checksums such as Cyclic redundancy codes (CRCs) used by many systems only provide protection against accidental or non-malicious errors on communication channels. As checksums do not make use of secret keys they can be rebuilt by malicious attackers. In order to be able to detect any of the aforementioned attacks usually integrity protection mechanisms based on cryptographic primitives are used.

Beside cryptographic integrity protection mechanisms based on asymmetric or symmetric cryptographic techniques efficient integrity protection can also be ensured by non-cryptographic techniques based on redundancy (see Section 2.19). In order to protect against message replay time variant parameters including random numbers, sequence numbers, and timestamps can be used. Sequence numbers can also be used to detect the deletion of messages. Thus they allow protecting the integrity of a whole sequence of messages [23].
Symmetric integrity protection techniques make use of the same key for generating and validating a checksum also known as Message Authentication Code (MAC). They either use a hash function or a symmetric encryption algorithm. Only with knowledge of the secret key it is possible to generate the cryptographic check value and to verify its correctness. Thus, all validators need to have access to the same secret key. Example for the checksum is the hash-based message authentication code (HMAC) [22] and the cipher block chaining message authentication code (CBC-MAC) [20] which makes use of a symmetric encryption algorithm (also see building block encryption).

Asymmetric encryption algorithms are used for generating digital signatures. First a cryptographic fingerprint of the data to be signed is generated using a hash function [24]. The hash value is subsequently encrypted using the private key of the signer. The verifier uses the public key of the signer to decrypt the signature. Then he compares the signed fingerprint of the message’s hash value with the fingerprint he generates using the received message. If both values match the signature has been verified and the integrity of the data has not been violated.

Integrity protection based on data replication in time or space can only be used if one can ensure that only a limited number of data replicas are altered by an attacker. The soundness of data is validated by comparing the available replicas. If a mismatch between replicas is found, the integrity of the data has been violated. If full replicas exist data can be reconstructed based on the unchanged copies with a defined lower bound for matching copies. More information on data replication is given in Section 2.19.

2.6.3 Effects on safety
Integrity protection mechanisms add an additional processing delay. This needs to be taken into consideration when designing time critical systems. For commonly used mechanism it can generally be said that the highest delay is added by signature generation followed by those using symmetric encryption algorithms and those using hash functions.

The undetected modification or deletion of data can have serious impact on safety, e.g. when sensor data are altered by an attacker in order to veil a critical system status or, when bogus sensor data raise false alarms.

2.6.4 Effects on security
Integrity protection mechanisms improve security and are required in any critical systems where data integrity threats exist.

2.6.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Delay</td>
<td>Type of mechanism dependent level of security</td>
</tr>
<tr>
<td></td>
<td>Delay</td>
<td>Key length dependent level of security</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synergies</td>
<td>Cryptographic checksums for</td>
<td></td>
</tr>
</tbody>
</table>
2.6.6 Interfaces
Message Integrity using cryptographic mechanisms

Generation
- Input:
  - Message
  - Nonce
  - Symmetric Key or Public Key
- Output:
  - Message Authentication Code or Signature

Validation
- Input:
  - Symmetric Key or Private Key
  - Message Authentication Code or Signature
  - Message
- Output:
  - Message
  - Right/Wrong

2.6.7 Application notes
Using digital signatures for ensuring message integrity provides not only the feature of message authentication but also data origin authentication. The term message authentication code implies that integrity protection mechanisms also provide some way of authentication. Message authentication ensures that data was sent by the trustworthy source (data origin authentication), and that it has not been altered (data integrity). In fact data which has been altered by e.g. a malicious attacker has a new source. Digital signatures are generated using a private key only known to the signer. Thus, the signature can be unambiguously assigned to the originating entity.

2.7 BUILDING BLOCK: CHECKSUMS

2.7.1 Name
Checksums

2.7.2 Description
In their most basic form, checksums are a kind of a redundancy check which allows the end user to protect the integrity of the system against accidental errors. Intentional manipulation is not in focus. Because transmission errors are inevitable, embedded protocols often utilize some form of a checksum to detect such errors. Checksums are also used to ensure data integrity of stored data for example in memories or on computer hard disks.
The data producing component typically calculates the checksum to be attached to the data. It is the job of the receiver to verify that the checksum matches with the data and the original checksum. The most basic arithmetic checksum algorithms entail adding up components or patterns in a message and then comparing these totals to the checksum value of the original message. Obviously this trivial mechanism does not protect against permutation errors. More advanced algorithms, which are based on polynomial division, like CRC [19] (Cyclic Redundancy Check), can detect more types of errors. The even more advanced ECC (Error Correcting Code) algorithms are capable to detect and correct errors by adding redundant information to the transmitted checksum. Often there is a three-way trade-off between the computing power used on the checksum, the size of the checksum itself and the probability of not detecting an error.

2.7.3 Effects on safety

Advantages:

- Basic checksums avoid system failures due to accidental data corruption
- Error correcting checksums harden data transmission and data storage by adding redundancy

Disadvantages:

- Additional components for checksum generation and check must fulfil certain SIL requirements since they can malfunction and decrease overall system safety.
  E.g. ISO 26262 states:
  "This requirement applies to ASILs (A), (B), C, and D, in accordance with 4.3: The development of safety mechanisms that prevent dual point faults from being latent shall comply with:
  a) ASIL B for technical safety requirements assigned ASIL D;
  b) ASIL A for technical safety requirements assigned ASIL B and ASIL C; and
  c) engineering judgement for technical safety requirements assigned ASIL A"

- Additional delay is introduced which has negative influence on WCET (worst case execution time) in real time systems.

2.7.4 Effects on security

Disadvantages:

- Error Detection Codes and Error Correction Codes are not intended to protect against intentional tampering and must not be used for security purposes.

2.7.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Higher computation effort spent on the checksum algorithm decreases the probability of undetected errors</td>
</tr>
<tr>
<td></td>
<td>Bandwidth demand increases because of the transmitted checksum.</td>
</tr>
<tr>
<td>Synergies</td>
<td>no effect on security</td>
</tr>
</tbody>
</table>
2.7.6 Interfaces

Sender:

- Input:
  - Message
  - Checksum algorithm
- Output:
  - Message
  - Checksum

Receiver:

- Input:
  - Message
  - Checksum algorithm
  - Checksum
- Output:
  - Message
  - Right/Wrong

2.7.7 Application notes

It is important consider that checksums like CRC and ECC are not suitable to achieve a higher security level. This mechanism only protects against accidental errors.

2.8 BUILDING BLOCK: BOOTCHECKS

2.8.1 Name

It is possible to understand “boot checks” as:

- Integrity check of the boot software
- Integrity checks done during the boot phase of the software.

In this document, the building block “Bootchecks” is defined as to be the integrity check of the boot software. Checks done during the boot phase of the software could contain any kind of checks e.g. SW configuration checks. “Secure boot” can be a type of boot check.

2.8.2 Description

The aim of the boot check is to verify the authenticity of the startup software or of the operating system or of the whole software. It ensures that this software part has not been modified by an unauthorized person. Depending on the software release process (i.e. conditions for the implementation of the signature), this check can also be used as a confirmation that the software release process has been successfully performed. So the boot check can be used for security, safety, quality, regulation and warranty reasons.

The boot check either checks the whole software or helps to build a chain of trust starting from the startup code up to the rest of the software.
In order to be effective, the boot check has to be started from a code located in a ROM and the code of the boot check can be located either in a ROM or be protected against modification by other means. Depending on the system requirements (e.g. on timing and security), the boot check process must either be finished before the startup code can be executed (foreground) or can run in parallel to the startup code (background). To be effective, the boot code in ROM has to take some measures when the boot check failed; the measures can be e.g.:

- No start of the operating system
- Reduced access to keys or security functionalities
- No debug access

In real time embedded software the timing constraint of such a check is very high since (in the case of the foreground version) it will delay the start of the system. Here a tradeoff between timing and security may be necessary e.g. (a)symmetric cipher / Secure Hash Algorithm / MAC based on AES / ...

2.8.3 Effects on safety
In case of foreground check:
The delay of the system start has to be taken into consideration. Depending on the safety goals and the safe state of the system, an execution time of the boot check above the planned budget could lead to the violation of a safety goal and to an unsafe state. So the...
boot check may inherit a timing requirement with the same safety integrity level as the application software.

If the hardware component storing the application software does not provide a built-in error correction code (ECC), then the boot check is a very strong method to check that the software has not been corrupted before start. But replacing the ECC or CRC with the authentication check would not allow anymore using the correction capability of ECC and differentiating between random hardware errors and attacks.

Measures in case of failed boot check:
The measures in case of failed boot check are driven from a security point of view but have also to be evaluated against the safety goals and the safe state of the system. If the measures are interfering with or are related to any safety requirements, then the measures will inherit the safety integrity level of the corresponding safety requirements.

2.8.4 Effects on security
The effect of the boot check from security point of view is the confidence in the authenticity of the software before start and/or during runtime (depending on attack scenarios). Hence further security analysis can rely on the authenticity of the executed software at system startup (or check the status of the software at startup time).

2.8.5 Trade-offs or synergies for safety and security
It can be plain text or a table for better comparisons of trade-offs and synergies, e.g.

<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trade-offs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Background check:</strong></td>
<td><strong>Background check:</strong></td>
</tr>
<tr>
<td>no time interference from boot check to the customer startup. No safety integrity level on boot check.</td>
<td>Lower confidence in the authenticity of the software at startup (depending on attack scenarios)</td>
</tr>
<tr>
<td><strong>MAC algorithm:</strong></td>
<td><strong>MAC algorithm:</strong></td>
</tr>
<tr>
<td>different algorithms take more or less computation time. This has to be considered on system level (not only for safety properties).</td>
<td>The more resistant against attacks the better.</td>
</tr>
<tr>
<td><strong>Foregroud check:</strong></td>
<td><strong>Foregroud check:</strong></td>
</tr>
<tr>
<td>The boot check must be developed according to the application safety level (if applicable for the system or no other safety measure available to limit time interference during startup)</td>
<td>Better confidence in the authenticity of the startup software.</td>
</tr>
<tr>
<td>Measures against <strong>unauthenticated software:</strong></td>
<td>Measures against <strong>unauthenticated software:</strong></td>
</tr>
<tr>
<td>The measures can affect safety functionalities. Therefore, the whole boot</td>
<td>The system shall reduce or disable some of or all the system function-</td>
</tr>
</tbody>
</table>
check or the part related to the safety functionalities must be developed according to the application safety integrity level.

<table>
<thead>
<tr>
<th>Synergies</th>
<th>Confirmations that the application software was developed and released according to the software development process. <strong>Liability</strong> issue of safety software for the software company.</th>
<th>High confidence in the software authenticity. Basis for further security considerations.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Detection of modified software for <strong>warranty</strong> issue.</td>
<td></td>
</tr>
</tbody>
</table>

### 2.8.6 Interfaces
- **Input:**
  - Startup code, operating system or application software containing a signature
  - Key / secret
- **Output:**
  - Confidence in the software authenticity
- **Prerequisites:**
  - Boot check code in ROM or trustworthy location
  - The information required for the signature generation need to be known also in the boot check. E.g. key exchange and storage mechanism.
  - Code of the boot check must be compliant to safety integrity level (system dependent)

### 2.9 BUILDING BLOCK: SW CONFIGURATION CHECKS

#### 2.9.1 Name
Software configuration checks

Check if the software configuration fits with the deployed software and hardware system (e.g. reprogrammable ECU with its microprocessor as used in automotive industry).

#### 2.9.2 Description
The software configuration check is based on predefined rules that need to be checked. This check is run on startup, before the application software is running. The software configuration can be seen as:

- One configuration set for the whole software. The startup sequence can choose between different configuration sets, or.
- One configuration item for each software module. The startup sequence can choose between different configuration items for each software module.
The aim of the software configuration check is to ensure that the chosen configuration:

- is consistent between the configuration items
- fits with the software that is going to execute
- fits with the whole system configuration

Configuration checks could also run in background for security reasons (depending on attack scenarios).

Compatibility of the software configuration items:
This system specific check aims to check the compatibility of some configuration parameter between configuration items. The necessity of such checks can be based on critical functionality, security or safety properties that need to be configured properly.

Compatibility check between software and software configuration:
This kind of check can be based on checksum or hash exchanged between the software and the configuration. The configuration can e.g. contain the hash of the software part meaning “this configuration can be used with this software”.

Compatibility between the configuration and available HW parts (internal or external e.g. communication device, sensors and actuators):
This requires to link specific configuration parameter to the availability of specific HW parts. The way how to check the availability of HW parts is system specific. The HW availability check can include the check of the HW parts authenticity.

2.9.3 Effects on safety
Any check added during the startup phase has to consider following safety aspects:

- Time and spatial interference:
  Either the configuration check is developed according to the highest safety integrity level of the application software or appropriate safety measures have to be defined so that this piece of software does not interfere with the safety properties of the system

- Software measures in case of failed configuration check:
  If the software measures in case of a failed configuration check are affecting safety features, then this part of the software need to be developed according to the safety integrity level of the affected features.

The compatibility check between software and software configuration allows ensuring that the combination of both elements successfully went through the software release process. As for the boot check, this allows limiting the system provider’s liability and warranty in case the compatibility check failed. Similarly, the HW parts authentication and configuration item check aims the same purpose.

2.9.4 Effects on security
The aim of the software configuration check from a security point of view is to detect attacks consisting in replacing HW parts, modifying the software configuration or modifying the software during runtime.
If the compatibility check between software and software configuration wants to be used for security purpose, then the hash needs to be replaced with a MAC (Message Authentication Code, see chapter 2.6). The reason for using a MAC is that, in contrary to a hash code, the value of a secret key has to be known in order to generate the correct MAC. Hence, an attacker cannot simply modify the configuration and generate the corresponding correct MAC since the key value is a secret unknown from the attacker.

### 2.9.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trade-offs</strong></td>
<td>The software configuration check may interfere with safety related properties (e.g. time requirements, software resources). If yes, the software configuration check must be developed according to the application safety level (if no other safety measure is available to provide a freedom from interference). This is related to higher development costs.</td>
</tr>
<tr>
<td>Measures against failed software configuration check:</td>
<td>Measures against failed software configuration check:</td>
</tr>
<tr>
<td>Measures can affect safety functionalities, the configuration check may be developed according to the application safety integrity level.</td>
<td>The system shall reduce or disable some or all of the system functionalities to block additional attacks, or avoid the divulgation of secrets.</td>
</tr>
<tr>
<td><strong>Synergies</strong></td>
<td>Check to reduce/confirm the system provider’s warranty and liability concerning the software configuration.</td>
</tr>
</tbody>
</table>

### 2.9.6 Interfaces

- **Input:**
  - Information about the software configuration (with signature if applicable)
  - Information about the software that is going to run
  - Information about the available HW parts
- **Output:**
  - Confidence in the authenticity and compatibility of the software configuration with the software/hardware system
- **Prerequisites:**
  - Code in ROM or trustworthy code
  - The information required for the signature generation need to be known also in the software configuration check, e.g. key exchange and storage mechanism.
2.10 BUILDING BLOCK: RUN-TIME MONITORING

2.10.1 Name
Run-time/online monitoring

2.10.2 Description
In safety-critical embedded systems, significant effort is spent on ensuring that the developed systems are reliable, safe and secure. Nevertheless there is no method that can guarantee, a priori, the absence of faults or vulnerabilities in the developed system, or eliminate the threats that may endanger the system at runtime. For this purpose run-time monitoring systems are required to ensure that deviations from the systems’ safety or security requirements are detected and treated properly.

In general purpose systems, where the real-time requirements and resource constrains may be less stringent than in embedded systems, the area of runtime monitoring systems (for reliability, safety or security) has been well studied. The following references provide taxonomies [46], and reviews [51] of the area. There has also been recent work in the area of runtime monitors for embedded systems [53], [52] though this work does not explicitly consider security aspects of the monitoring or the tradeoffs that must be made between safety and security.

For clarity, we will provide a brief definition of what we mean by monitoring systems, partially based on [47]: “the behaviour of the system is observed and monitoring information is gathered; this information is used to make management decisions and perform the appropriate control actions on the system”. In general, when we deal with the monitoring of computing systems we have one or more monitored systems (e.g., computers, operating systems, embedded application) that send out signals or measurements of their well-being or distress to a monitoring site (logically centralised, even when computing elements are distributed), which thus can extract (or help humans to extract) various forms of processed information and make (or help humans to make) decisions. As stated in [53], observing the monitored elements execution can be done at runtime, and the results of the observations can be examined subsequently offline. This is known as offline monitoring; whereas when observed data are examined at runtime, the task is referred to as online or runtime monitoring. In this building block we will refer to runtime monitoring of software execution unless otherwise qualified. We will also use the terms monitors and monitoring systems interchangeably.

In terms of the actions that a monitoring system takes when receiving the signals, they can be classified under interventionist or non-interventionist monitoring systems. An example of a non-interventionist monitoring system is an Intrusion Detection System (a system which only raises an alarm about a suspected intrusion, but does not actively prevent it), whereas an example of an interventionist system is an Intrusion Prevention System (which detects as well as takes some preventive action on an attempted intrusion).

Monitoring systems are being increasingly proposed and used for embedded systems also [53]. As manufacturers tend to embed “telemetry” capabilities to monitor the performance and health of the hardware which the computers control (from engines in aircrafts, trucks and cars to factory equipment and house appliances) and support maintenance and repair, so they can add similar capabilities to monitor the performance and health of the embedded systems. Like other monitoring practices,
these can be used to evaluate the safety and security of these systems. A sort of online monitoring is frequently employed for error detection, e.g., as in “watchdog” technique, where a small coprocessor is used to monitor the behaviour of the main processor.

In terms of where in the embedded system architecture the monitoring systems may be deployed [52] provides the following classification: hardware based monitors (e.g. embedded directly on the chip), software based monitors, or a hybrid of the two approaches. An important consideration in embedded systems is the resource sharing between the monitored system and the monitor.

Focusing on prevention monitoring, especially from the security perspective, we refer to such monitoring as enforcement mechanism. Referring to [50], an enforcement mechanism acts by controlling system’s action step by step in order to guarantee that a required security policy is satisfied by all the actions performed by the system. As an example, let us consider the e-health use-case. Since the data involved contain sensitive information, mechanisms of data protection that prevent lack of information are necessary for guaranteeing security and privacy.

An enforcement mechanism works by following the execution of the considered system. Whenever a bad action, i.e., an action that can cause a violation of requirements, is going to be performed, the mechanism reacts in order to prevent the violation. It can be done following several strategies: the execution can be stopped, or the bad action can be corrected by the insertion or suppression of some action. In this way the execution trace is modified in order to obtain an execution trace that satisfies the considered requirements.

From the practical point of view, these strategies can be implemented in different ways. We propose two different implementations; one that is able to guarantee access control policies written in eXtensible Access Control Markup Language (XACML) and another that is able to manage behavioural requirements, in order to manage usage control policies, access control policies and security policies.

2.10.3 Effects on safety
Monitors are clearly used to enable timely detection and response to failures and incidents. So from that viewpoint they benefit the overall system safety. However in embedded systems that may have stringent real time or resource requirements, the deployment of monitors needs careful consideration to avoid instances where a monitor prevents a timely execution of a function that may have safety repercussions. This is especially true in cases of interventionist monitors, but holds even for non-interventionist monitors that are sharing resources with the monitored system and may deprive a timely access to a resource needed by a function of the monitored system.

Adding a monitor to an embedded system will increase its overall complexity and hence may impose more complex certification requirements on the overall system.

2.10.4 Effects on security
The effects on security are similar to those on safety. Additionally, the monitor itself may be a source of vulnerabilities and hence susceptible to attacks. If the monitor is compromised and remains “silent” when an attack on the monitored system takes place, then the safety of the monitored system may be compromised too.
2.10.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Variable and trade-offs</th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introducing monitoring</td>
<td>...improves the overall</td>
<td>...but may give the attacker a larger “attack surface” of the system: i.e. the monitor itself becomes a source of vulnerabilities and attacks.</td>
</tr>
<tr>
<td>systems ...</td>
<td>safety of the system because the safety incidents are more likely to be detected and prevented</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introducing monitoring</td>
<td>...reduces the overall</td>
<td>.. but it may increase security as it will allow detection and possibly prevention of security attacks on the monitored system by the monitor.</td>
</tr>
<tr>
<td>systems ...</td>
<td>safety because of resource contention between the monitoring system and the monitored system which, for example, may cause real-time deadlines to be missed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synergies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introducing monitoring</td>
<td>...reduces likelihood of safety incidents and failures due to accidental errors remaining undetected and hence not prevented</td>
<td>... as well as the same failures remaining undetected when caused by malicious attacks</td>
</tr>
<tr>
<td>systems ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.10.6 Interfaces

- **Input:**
  - The parameters, behaviors, events, or functions of the monitored system that need monitoring
  - The requirements: in particular, security and safety requirements that the system has to satisfy.
  - Safety/Security policies
- **Output:**
  - For example the “alarm” or other “informative” monitor output (for non-interventionist monitors) or preventive or corrective action on the monitored system (for interventionist monitors)

2.10.7 Application notes

We expect to use the Avionics use case supplied by EADS to illustrate how monitors can be applied with embedded systems.

We also aim to use the e-health use case for dealing with security and privacy aspects.
2.11 BUILDING BLOCK: PLAUDIBILITY CHECKS

2.11.1 Name
Plausibility Checks

2.11.2 Description
Plausibility checks are a mechanism generally used to validate the correctness of data. Such data can be input data (e.g. stemming from signal sources) as well as output data (e.g. the data values computed by a controller based on input data to trigger an actuator). Plausibility checks can even be used to validate inputs provided by a user or operator of the system.

With plausibility checks it is possible to detect typical value related failures like wrong value, stuck value, value out of range etc. The general strategy is to check the value(s) in question against predefined values (e.g. a range check) or with values stemming from different source or computation methods. Some examples for commonly applied plausibility checks are provided in [29]:

- A range check for e.g. speed or steering angle. If not in a given range, the value must be incorrect.

- A double check with some other value which must be in a certain proportion to each other due to physical reasons: speed vs. wheel rotation, vehicle ignition state and engine torque etc.

- Using the same input values to compute the same output values but with the diverse implementation/duplicate hardware and compare the results.

Sometimes, a potential strategy for the usage of a plausibility check is to trigger the system with some artificial created value for which the expected output is known and then compare the system output with the expected output [30].

2.11.3 Effects on safety
Plausibility checks are widely used in safety related systems to detect failures and to trigger appropriate measures against such failures – e.g. move to a safe state. A plausibility check is usually implemented by an additional logic which requires additional hard and/or software as well as computation power. So it usually consumes system resources. On the other hand, without plausibility checks system safety is hardly to be achieved.

2.11.4 Effects on security
Given the fact that failures may be injected into the system by security attacks, plausibility checks become even more important to protect the system and to operate it in a safe mode. Plausibility checks can in that sense being used as a mechanism to validate data which may be suffered from security attacks (e.g. manipulated).

On the other hand, plausibility checks used for safety shall be analyzed for potential security hazards that may influence the result of the plausibility check. If e.g. two signals are compared with each other, it has to be ensured, that the values are not manipulated. In that sense, plausibility checks should be applied in a secured environment.
2.11.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Plausibility checks usually consume system resources. If used as a mechanism to validate data which may be suffered from security attacks, this is to be considered as additional effort and may influence time critical safety related functionality.</td>
<td>In order to have an effective plausibility check, it has to be ensured, that the data which is used to compare with is available and not manipulated. Otherwise the plausibility check may lead to wrong results.</td>
</tr>
<tr>
<td>Synergies</td>
<td>Commonly applied mechanism for increasing confidence in the data used in the system</td>
<td>May increase confidence in data that stems from outside sources and may be manipulated.</td>
</tr>
</tbody>
</table>

2.11.6 Interfaces

Since there are multiple usage scenarios for plausibility checks, there is no common interface how a plausibility check can be used as a mechanism or building block within a system. The concrete implementation of a plausibility check depends on the values that need to be checked for plausibility.

A very abstract view could be that the plausibility check takes one or more signals as input and produces as output a logical value declaring the input as plausible or not plausible. It can further produce a value that is plausible, e.g. by extrapolation/interpolation which can be used to further operate the system. This might be necessary, if a transition to a safe state is not possible and the system must continue to operate for some time.

In [28] a general model for a plausibility checking system usable in automotive is given as follows:
D2.1 Specification of Safety and Security Mechanisms

As to be seen, the input signals are checked in 3 potential manners based on the concrete case:

- Observing a single value – e.g. to be in the specified range
- Observing multiple values that are produced or obtained from redundant sources
- Checking values to be valid in a certain model – e.g. consistent in an expected physical function.

The output of these checks is forwarded to an evaluation engine, where a weight procedure is applied to make the final decision on plausibility.

In summary, the following generic description of the interface can be applied:

*Input*

- **Signal to check for plausibility (mandatory)**
- **Additional signals (optional)**

*Output*

- **Value to specify the confidence in the signal (mandatory)**
- **Plausible value (if extrapolation is used (optional)**

2.12 BUILDING BLOCK: LOGGING

2.12.1 Name

Logging

---

1 Figure taken from [28]
2.12.2 Description
Logging is a widely used mechanism in a variety of software systems in order to produce a continuous report of events that occur during the runtime of a system. These reports are usually called “log files” and according to [31] they are used for debugging, fault location, regression testing and administrative information.

The typical nature of how a software system uses a log file is as follows:

- The log file is a separate output of the system, usually as ASCII text.
- The log file is empty at the startup of the system or it still contains some information from a previous run. During runtime the system appends additional lines to the log file, but never deletes or changes any previous information.
- Every line or group of line reports on a specific event like an input or output, receiving or sending of a message, the parameters or results of a function call, the setting of variables, or the current values of variables.
- The reported information has to be specified during the development of the system and should comply with the requirements of a proper monitoring of the system and/or the detection of faults.

2.12.3 Effects on safety
Logging alone might not help to reduce the risk of a hazard in a safety related system. But in combination with programmatic methods for log file analysis and also in combination with other building blocks, like plausibility checks, it can help to detect failures and to activate appropriate correction mechanisms. Of course, as for other building blocks, the creation of log files and their analysis needs to be implemented as an addition to the system’s behaviour implementation and therefore it consumes computation power and even more critically, depending on the amount of reported events, it consumes memory space.

2.12.4 Effects on security
Logging and an appropriate log analysis method can be good measures to detect security attacks to the monitored system. For instance, if a function requires the authentication of the user and if the logging reports the corresponding login events then the log file also outlines unfriendly login attacks. In case the log file is input for a more safety related building block, like plausibility check, it has to be assured that the log file itself was not the target of a security attack. This might be achieved by the usage of an encryption and/or signature generation building block, for instance.

2.12.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Logging consumes system resources. If used as a mechanism to provide data for</td>
<td>In case the log files are used to feed other building blocks it has to be assured that the</td>
</tr>
</tbody>
</table>
Synergies

<table>
<thead>
<tr>
<th>Other Safety Related Building Blocks, This Is To Be Considered As Additional Effort And May Influence Time Critical Safety Related Functionality.</th>
<th>Log File Itself Is Not Manipulated By Security Attacks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could Provide Input Data For Other Safety Related Building Blocks, E.g. Plausibility Checks Or Boot Checks</td>
<td>Commonly Applied Mechanism To Monitor Security Attacks.</td>
</tr>
</tbody>
</table>

### 2.12.6 Interfaces

The Logging building block usually has well-defined system events as input data. These events are then formatted and stored to log file. The log file can be analyzed later on and could feed other building blocks. The following picture illustrates these interactions.

![Figure 2-5: Interactions for the Logging building block](image)

2.13 Building Block Security Audit

2.13.1 Name

Security audit

2.13.2 Description

Security audit is a feature, which, when enabled, documents all information flows between system components and/or external environment. Security audit can be split into

a. generation of audit logs,

b. export or storage of audit data, and

c. (optional) off-line analysis of audit data.
Note: For completeness, we mention support for off-line analysis. Such an analysis can be product specific, and therefore, should be developed as part of a particular project. If a generic approach for analysis can be defined, the list of the SESAMO BB can be revised.

A security audit shall be able to record all security events such as all information flows. For example, security audit has to audit all deny/allow decisions for a given flow.

Audit of all deny decisions can be used to

a. Detect abnormal behavior
b. Detect attempts of security breaches

All allow decisions can be used to

a. Prove that recipient of an information received it (in computer security jargon, this feature is called “non-repudiation”).
b. Detect security breaches, e.g.
   i. After a strike of a cosmic particle there was an unintended bit-flip in a rule in an access control matrix. Thus, this information flow is allowed. All application of that rule has to be recorded to be able to detect security breach.
   ii. Due to a bit flip that was triggered by a malicious user via physical access, plugged devices, or intentional irradiation.

2.13.3 Effects on safety
Depending on the level of “Security Audit”, this BB can create a log entry for every security relevant event. This can significantly decrease the overall system performance. Therefore, the level of security audit has to be selected carefully with simultaneous assessment of the system safety.

2.13.4 Effects on security

2.13.5 Trade-offs or synergies for safety and security
“Safety Monitoring” (or health monitoring) has different goals than “Security Audit”. The main difference is that

- “Safety Monitoring” tries to detect abnormal behavior and tries to keep system in health state. For example: on detected critical events “safety monitoring” will try to correct the system to keep it running safely whereas “Security Audit” will generate logs which could result in full stop of the system to prevent any non-authorized information flows. Full stop for aircraft in the air can be fatal.
- “Security audit” is monitoring security relevant actions and generating corresponding entries for analysis. The reaction on security relevant events is not necessary part of a security audit engine. Usually, reaction is implemented in a domain/product specific context.
Trade-offs | Possible performance degradation for a system with detailed audit | Full control of all security relevant actions, e.g. logging all positive and negative decisions
---|---|---
Synergies | Security Audit can enhance system safety, e.g. reaction on security breach can prevent safety failure | Security Audit can enhance system security, e.g. reaction on safety failure can prevent creating vulnerability, and hence, prevent an exploit.

### 2.13.6 Interfaces

- **Input:**
  - List of security relevant events
  - Audit mode or level of details for audit records
  - Periodic checks (optional)
- **Output:**
  - Audit records for every configured event

### 2.13.7 Application notes

### 2.14 BUILDING BLOCK: LEVELS OF OPERATION

#### 2.14.1 Name

Levels of Operation

#### 2.14.2 Description

Systems are often designed along the idea that they usually operate normally but with occasional abnormal operation.

When security is involved it is a much better design principle to determine right from the start several levels of operation and

- the expectations of the system at each level, and
- the procedures for changing between levels.

Normal operation and total failure would then be two extreme levels of operation.

The need to change between levels of operation may be due to hardware and software faults relating to safety issues but may also be due to security attacks where parts of the system gets compromised or where denial of service attacks makes it impossible to properly authenticate attempts to operate the system.
2.14.3 Effects on safety
When faults occur the level of operation changes and for safety one often has different notions of fail-safe error modes and fail-operational behaviour. The challenge is to deal with as many faults as possible and still be in control of a safe way to operate the system (perhaps by stopping it in a safe way if this is feasible).

2.14.4 Effects on security
Security policies describe restrictions on who is allowed to control a given system. In an emergency there may be a need to override such standard policies. Mechanisms should be designed for how to bypass security policies in a way that it is still considered acceptable.

For classical security policies it often works to log accesses that violate the policy because legal punishment may be sufficiently strong to incite people not to misuse the overriding possibilities.

The change between different levels of operation is often a sore point in the design of security models – many shortcomings of existing policies have been identified in this phase, thereby highlighting the importance of dealing with it up front. In particular, the classical weaknesses of mandatory access control systems like the Bell-LaPadula system, relate to the lack of policies for how to change between different levels of operation as embodied by the access control policies enforced.

2.14.5 Trade-offs or synergies for safety and security
Security has a number of important facets that classically can be summarized as follows:

- **Confidentiality (or privacy):** that data do not get in the hands of those that should not have it.
- **Integrity (or authenticcy):** that data (or control) does not get modified by those not authorized to do so.
- **Availability:** that resources are always available.

Safety also has number of facets describing how to avoid that no harm comes from using the system and a particular challenge is that of failure which is fairly comparable to availability in terms of the challenges offered.

To assess the interplay between safety and security it is important to understand the many facetted nature of security as explained above, because it will not suffice to merely say that “in an emergency safety takes precedence over security”.

Rather it may be prudent to say that

“(in an emergency) safety takes precedence over confidentiality”

because in an emergency (and perhaps in general) it would not be prudent to abstain from taking safety measures on the grounds that privacy of people could be violated. Similarly, it is prudent to say that

“even in an emergency, integrity takes precedence over safety”

because integrity (in particular authenticity) is what ensures that control actions originate from those skilled to do so. As an example, if an emergency happens in a plane at 30,000 feet it is still essential that corrective actions must come from the cockpit and not from malicious software on a passenger tablet connected to the intranet in the plane.
Levels of operation provide the proper frame for determining when to relinquish normal security policies in the case of emergency thereby allowing to bypass some of the authenticity measures if disaster is imminent.

2.14.6 Interfaces
- **Input:**
  - The failures that seem relevant to consider for the safety goals
  - The confidentiality, integrity and denial-of-service considerations that seem relevant for the security goals
  - Considerations of situations where one may need to deviate from the ideal goals
- **Output:**
  - Safety and security policies at each level of operation
  - Policies for when to change level of operation

2.14.7 Application notes
Key notions from safety and security can be illustrated in the following way where we compare the IMA requirements from safety with the MILS requirements from security.

<table>
<thead>
<tr>
<th>IMA Requirements</th>
<th>MILS Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time and space partitioning</td>
<td>Separation</td>
</tr>
<tr>
<td>Intra/inter-partition communication</td>
<td>Controlled information flow</td>
</tr>
<tr>
<td>Mixed safety criticality levels</td>
<td>Multiple levels of security (trusted code, untrusted code, etc.)</td>
</tr>
<tr>
<td>The duty of the partitioning kernel and</td>
<td>The duty of the separation kernel and of the application designer</td>
</tr>
<tr>
<td>of the programmer</td>
<td></td>
</tr>
<tr>
<td>Replication and spare modules</td>
<td>Non-by-passable and tamper-proof</td>
</tr>
</tbody>
</table>

**Table 2-5: Safety and security key notions**

<table>
<thead>
<tr>
<th></th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>A careful consideration of the limited functionality that needs to be imposed due to potential faults.</td>
<td>A careful consideration of the limited functionality that needs to be imposed due to potential attacks on the system.</td>
</tr>
<tr>
<td>Synergies</td>
<td>A better understanding of the extent to which safety can be guaranteed at the various levels of operation.</td>
<td>A better understanding of the extent to which security risks may need to be taken in order to continue overall operation</td>
</tr>
</tbody>
</table>
2.15 BUILDING BLOCK: INFORMATION FLOW CONTROL

2.15.1 Name
Information Flow Control.

2.15.2 Description
Access Control is determining who has the right to observe or modify data in the system; observation mainly deals with issues of confidentiality and privacy whereas modification mainly deals with issues of integrity and authenticity. The outcome of the decision is usually simply a "yes, the operation will be admitted" or a "no, the operation will be blocked".

Information Flow Control goes beyond access control in tracking how information propagates through a program during execution to make sure that the program handles the information securely. This may be linked to access control in that certain accesses are only allowed if it can be shown that future use is somehow restricted (e.g., the passenger list can be obtained if only the number of passengers is used rather than information about their ethnicity) or if the history of producing the data is sufficiently trustworthy (e.g., the weather data used for a course change can be shown not to have been corrupted by third party, perhaps due to the disclosure of cryptographic keys).

Information flow control is vital for large-scale systems, where it is not trivial to prevent passing data to code that is not trusted. As the trust requirement is transitive, i.e. any code the data might travel to must also be trusted, complete trust becomes unattainable as the systems grow larger and more complex.

2.15.3 Effects on safety
The propagation of sensitive data to untrusted parties may cause safety problems depending on the system. Especially important for systems where classified data is used in the software. As an example, on an airplane low quality information about the weather may be sufficient for informing the passengers but may be unsuitable for making decisions in the cockpit about changing the flight plan (including altitude and course).

2.15.4 Effects on security
Information Flow Control can provide guarantees about information propagation. Therefore, it addresses an important issue in security that none of the other building blocks such as cryptography, access control, etc. are dealing with.

A typical burden in the usage of the information flow control is the overhead on programming. The developers should be trained to use information flow constructors (e.g. when using a language that supports information flow control), and the analyses should not be too restrictive. The trade-off in this case is, either achieving higher level of security but sacrificing some of the valid flows or being content with a modest level of security but not risking valid flows to be labelled as invalid.
2.15.5 Trade-offs or synergies for safety and security
As can be expected from majority of the security mechanisms, a heavy implementation of information flow control may risk the fulfilment of the timing requirements thus leading to risks in safety. However, this is not an issue on the static implementations that take place before the execution.

2.15.6 Interfaces
- Input:
  - Source code
- Output:
  - Certification of the code, e.g. no flows violate the security policy.

2.15.7 Application notes
Information Flow Control is an MILS requirement in the Avionics use cases (EADS).

2.16 BUILDING BLOCK: PARTITIONING

2.16.1 Name
Partitioning

Also known as “Time Partitioning”, “Space partitioning”.

2.16.2 Description
Partitioning is used to help segregate mixed-criticality applications as well as to support the verification and certification process. For that purpose partitioning splits a set of resources into different usage domains. Every partition has assigned a distinctive criticality level. Some of the partitioning goals are to control how resources are used, to control interferences between, to share hardware and software resources such as CPU and network schedules and to isolate faults in the safety case and to encapsulate against attacks in the security case. The typical resources which are partitioned are time and space.

- Time
  We define time as computational time of the platform. For the rest of this subsection we interpret time as
  - CPU cycles
  - Bandwidth of busses, which are shared among CPUs, I/O devices, and other platform components.

- Space
  We define space as all resources of the platform. These resources are
  - Memory
  - I/O devices
  - Files
  - Buses
  - CPUs
2.16.3 Effects on safety

Advantages:
- For space partitioning: Protection of program, data, registers and dedicated I/O. Persistent storage / memory is writable by only one partition. Temporary storage locations / registers of a partition are saved when control is transferred and not contaminated.
- For time partitioning: Protection of processing time and communication bandwidth assigned to a partition. There is guaranteed access to a prescribed set of hardware resources for a prescribed period of time. The order of execution between communicating partitions is consistent for each execution frame, resulting in a defined execution of the threads contained in the correlated process of the partition, with every thread fully allocates its CPU, I/O, Files, Buses resources every period.
- In principle, partitions allow multi-criticality on shared resources.

Disadvantages:
- Overhead of any kind, also delays when context-switching,
- Hard to prove to keep guaranteed maximum latency thresholds taking into account all intra- and inter-partitions communication possibilities of a schedule policy,
- Since every partition has assigned a criticality level, all software in a partition has to be certified to that criticality level at least.

2.16.4 Effects on security

Advantages:
- Partitioning is compatible with the “Multiple Independent Levels of Security” concept:
  - Separation and controlled information flow,
  - implemented by separation mechanisms that support both untrusted and trustworthy components,
  - ensuring that the total security solution is non-bypassable,
  - ensuring that the total security solution is evaluable, and
  - always invoked and tamperproof.

Disadvantages:
- Security mechanisms include checks at start-up and checks during operation. The latter are the more interesting ones and they have to run in parallel to threads inside a partition. No security mechanisms are allowed to destroy any of the partitions principles. On the contrary: security has to be part of the overall Integrated Modular Avionics System whichever is made up of partitions and is subject to scheduling policies and timing principles. And all security related software in a partition has to be certified to the criticality level of that partition at least. An example is: A watch dog detecting intrusions in a level A partition has to be certified on level A or it has to run outside that partition.

2.16.5 Trade-offs or synergies for safety and security

<table>
<thead>
<tr>
<th>Trade-offs</th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance of partitions context switches</td>
<td>Attacks on context stored information</td>
<td></td>
</tr>
</tbody>
</table>
Segregation of functionalities into partitions and the intra-partition communication limited inter-partition communication partition I/O

The partitions principle provides a schema, not to be violated by any added safety and/or security principles and code.

Strong cyclic scheduling of partitions allows timing analysis as guaranteed latency thresholds, a WCET is existing

Security software has to be real time qualified (i.e. has to follow real-time requirements) and has to fit in strong cyclic scheduling of partitions; it is not allowed to destroy timing-analysis “principles”

All in a partition has at least to be certified to the partition level

All in a partition has at least to be certified to the partition level, that includes all security related software

Synergies

Similar principles:
Partitioning principle for safety

Similar principles:
Partitioning principle for security

2.16.6 Interfaces

- Input:
  - List of available resources
  - List of what can be partitioned per resource
  - List of applications with isolation, separation, or segregation requirements
- Output:
  - Domains containing partitioned resources
  - Assignment of applications to domains

2.16.7 Application notes

Partitioning can be implemented via different techniques.

- IT system design, e.g. by using multiple computers connected via networks
- On one system, e.g. domain specific requirements/standards such as ARINC 653
- Virtualisation can be used as a mechanism to implement partitioning

We identify the following important properties of partitioning

- Isolation – to isolate faults, virtual resources. This property has related to exclusive usage of partitioned resources.
• Separation – to separate information flows, interferences in the presence of shared resources.

2.17 BUILDING BLOCK: VIRTUALISATION

2.17.1 Name
Virtualisation

2.17.2 Description
Virtualisation is technique to share one physical resource among many resource consumers such that every consumer will have illusion of owning the resource exclusively. Thus, this technique allows building on the same physical resource several logical resources, which behave as new physical resource for the end user.

This building blocks considers virtualisation of the following resources available on a platform:

• CPU time
• Memory
• Devices
• Buses

2.17.3 Effects on safety
Advantages:
  o Segregation of different functionalities on one hardware

Disadvantages:
  o Determinism can affected due to the sharing of physical resource
    ▪ Example: sharing CPU can introduce various delays when switching contexts between virtual machines. If the switch time depends on the current state of the CPU, then the switch will depend on the computation in previously active virtual machine. Thus, it would be hard to create predictable/deterministic system design with hard and fin-grade real-time requirements.

  o Error propagations due to some specific of the virtualized physical resource
    ▪ Example: handling interrupts from devices in virtualized environment

2.17.4 Effects on security
Advantages:
  o Virtualization allows explicit information control via shared physical resources

Disadvantages:
  o New side channels can arise due to specifics of the virtualized resource

2.17.5 Trade-offs or synergies for safety and security
In this section we identify pairs of “safety” and “security”
<table>
<thead>
<tr>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trade-offs</strong></td>
<td><strong>Residual information flows through non-cleaning information for context switch</strong></td>
</tr>
<tr>
<td>Performance of virtualisation, i.e. speed of context switches</td>
<td>Side effects of segregation, e.g. side effects on device state by access from different virtual functions.</td>
</tr>
<tr>
<td>Efficient segregation of different functionalities</td>
<td>Dynamic time budgets, TDMA</td>
</tr>
<tr>
<td>Efficient segregation of different functionalities</td>
<td>Timing-analysis based attack, e.g. confidentiality for strict one-way communications</td>
</tr>
<tr>
<td>Dynamic time budgets, TDMA</td>
<td><strong>Synergies</strong></td>
</tr>
<tr>
<td>Static TDMA, predictable timing</td>
<td>Fault isolation</td>
</tr>
<tr>
<td>Static TDMA, predictable timing</td>
<td>Attacks based on fault injections</td>
</tr>
<tr>
<td>Separation of different safety criticalities</td>
<td>Separation of different security criticalities</td>
</tr>
</tbody>
</table>

### 2.17.6 Interfaces
- **Input:**
  - List of HW resources with support for virtualisation
  - List of resources with SW virtualisation only
    - These resources usually do not support virtualisation on HW levels but implement a “virtualisation manager” which allows usage of one physical resource in virtualised manner. This “virtualisation manager” implements the safety and security policies for virtualisation such as TDMA (time division multiple access), context switches, number of virtual functions etc.
  - List of virtual functions per resource with virtualisation support
    - Example: Virtual functions in Single Root IO of PCIe (PCI-SIG, 2010)
  - List of required virtual functions and segregation requirements on one virtualised resource
- **Output:**
  - List of instantiated virtual functions
  - Configurations for usage of virtualised resources, e.g. scheduling for TDMA
2.18 BUILDING BLOCK: PROTOCOLS FOR SECURE REAL-TIME COMMUNICATION

2.18.1 Name
Protocols for secure real-time communication

2.18.2 Description
One of the key properties required for real-time communication protocols is the possibility to calculate worst case behavior such as the worst case time needed for delivering a message from the source to the destination. In order to perform such a worst-case analysis the network must employ deterministic algorithms that grant access to shared resources. In case of systems with bus topology, the shared resource is the bus itself and deterministic medium access control algorithm is required. Another feature that is important in real-time networks is that the maximum length of transmitted messages is limited. Since the message transmission is not pre-emptible, even the highest criticality message has to wait until the previous transmission is finished. Limiting the message size therefore limits the “waiting” time for high criticality messages.

Both before mentioned properties can be demonstrated on Controller Area Network (CAN). It uses deterministic medium access control algorithm called Carrier-Sense, Multiple Access with Collision Resolution (CSMA/CR) and limits data payload to 8 bytes per message. This short limit represents a problem when one wants to add security features to a real-time communication protocol. One of the often requested security properties is message authenticity that allows a receiver to know the identity of the sender and prevents forging the sender’s identity. As described in Section 2.6, message authentication is often implemented by appending a message authentication code (MAC) to every message that needs to be authenticated. If the maximum message size is very limited, such as in the case of CAN, it is problematic to append MACs to messages. It is possible to send MACs in additional messages, but this increases the bandwidth required by the application and the system may not provide enough bandwidth for this. Also, having MACs in separate messages increases the time the receiver has to wait to receive the whole message. Alternatively, one can use shorter MACs that fit into the same message as the data. This is better from bandwidth point of view, but shorter MACs are less secure because they are easier to attack. For example, one can use brute force to quickly find the key used to generate MACs.

2.18.3 Effects on safety
Adding message authentication codes (MACs) to messages increases transmission times and requires more bandwidth. This may lead to missing deadlines for safety-related messages.

2.18.4 Effects on security
Adding MACs increases security as only the one who knows the proper secret key can communicate with other nodes in the system.

2.18.5 Trade-offs or synergies for safety and security
There is a clear trade-off between safety and security. Adding “too much” security may lead to missing deadlines, which negatively affects safety. System designers need a way how to find the best possible security properties for their system, while keeping it safe.

On the other hand, adding redundant data (MACs) to messages decreases the chance that corruptions in received messages will not be detected. In this regard, MACs are better than traditionally used CRCs and adding MACs is good for both safety and security.
2.18.6 Interfaces
Current schedulability analyses have to be extended to take the tradeoff between safety and security into account. The interfaces to the extended analyses could be as shown below. We consider the case for CAN bus, but similar algorithms could be developed for other networks.

- **Input:**
  - Communication requirements of individual nodes/applications/signals together with required temporal and security properties e.g. message sizes, communication periods, deadlines, security requirements (levels)

- **Output:**
  - Whether the system is schedulable (i.e. all deadlines are met).
  - In case of schedulable system, the communication parameters for individual applications/messages/signals, i.e. the priority of CAN frames, the length of MAC for different signals, how often to refresh session keys etc.

2.19 REDUNDANCY AND DIVERSITY

2.19.1 Name
Redundancy and diversity

2.19.2 Description
We call redundancy the use of more than one functional modules, all performing the same function (as seen from an abstract, system-level viewpoint), although possibly differing in how they accomplish it, to improve the likelihood that the function is performed correctly even if one of these fails. This is sometimes described as modular redundancy.\(^2\) There are many surveys of such architectures and some standards list specific options as recommended.

We call diversity, in a system built with redundancy, the intentional differentiation between the ways the redundant components are built. Because common defects in these redundant components may make them likely to fail together and thus reduce the intended effect of redundancy, diversity is applied to avoid (reduce the likelihood of) common failures due to systematic causes (digital de-

\(^2\) This definition can easily be interpreted to encompass very different uses of the principle of redundancy, extending for instance to the use of error correcting codes in memory (where the function that is protected is the storage of 1 bit, by using extra bits – possibly fewer than 2 for each bit of storage required at system level. This document uses a pragmatic classification of concrete building blocks that are of interest in SESAMO. This may lead to a concrete design element being in theory a member of two of the categories of building blocks defined, but this is not a problem for SESAMO.
sign defects common to identical replicas\textsuperscript{3} of the function, low reliability in certain ranges of environmental conditions, unknown aspects the controlled physical phenomena that may create specification defects in the protection / control systems).

For architectures with diversity, e.g. [73] is a concise description of the principles and the design space; [70] lists many specific options presented in the literature about software diversity. [77] surveys current applications in various industry sectors.

Once the function to be implemented is defined, a redundant design for it is characterised by at least these parameters:

- the number of redundant instances (replicas) of implementation of the function
- the deployment (software-hardware mapping) and scheduling of these redundant instances. In many safety critical applications, each replica of a function is statically allocated to a hardware element and executed unconditionally when the system-level function is required. But there are alternatives, e.g. a 3-redundant implementation of a software function in which 2 replicas execute at each invocation, and \textit{if and only if} they disagree the 3rd one is also executed as a tie-breaker. And in some cases, redundancy is only in the time domain: a computation is repeated (by the same code or different code) in more than one time slot, to tolerate transient hardware faults or application-level software faults
- the \textit{adjudication} process that decides what output is seen at system level when there are disagreements between the replicas. For instance, a 2-redundant system may work as 1-out-of-2 for some action (the action is performed if at least one of the two orders it) or 2-out-of-2 (the action is only performed if both order it). The adjudication function chosen depends on the desired trade-offs between dependability goals, e.g. safety vs availability
- any redundancy management scheme applied to deal with apparent permanent faults by reconfiguring the redundant system, e.g. to permanently de-activate a faulty instance.

Given a redundant design, the forms of diversity applied to it are defined by

- which ones of the redundant instances are diverse; e.g. a triple-redundant system may contain two instances that are identical copies and one that is different from them, or three different variants (cf the Space Shuttle flight control system with 4 identical replicas plus 1 diverse backup)
- how diversity is achieved; e.g. this may include purchasing from different companies; ensuring differences that are considered substantial among the required behaviour of the diverse instances; ensuring specific differences among the designs and development methods and tools applied; ensuring no overlap between the staff, components, libraries etc used in building the diverse instances; and other possibilities [59]

\textsuperscript{3} There is no standard term across industrial sectors for the replicated (possibly diverse) instances of a component or function that together form a redundant / diverse component. There are many industry and even system-specific names (“channels”, “lanes”, “trains”, etc). We will use \textit{replicas}. 
D2.1 Specification of Safety and Security Mechanisms

- it is useful to observe that “diversity” is not a single characteristic. Implementations of the same function may be different in many respects; specific differences will be useful against subsets of the possible causes of common failures but not all [59].

Diversity as a principle can be applied not only to the developed system but to the development process of any system or component. For instance, the benefits of diversity between V&V methods have been studied in some depth [56][57]

2.19.3 Effects on safety

Redundancy has the purpose of enabling a function to be performed correctly despite faults in the redundant components. The redundant design chosen determines what fault combinations are tolerated, which dependability attributes are improved and to some extent the quantitative improvement achieved. So, for instance:

- for a safety shut-down function, 1-out-of-2 redundancy improves the likelihood of the shut-down being performed when needed, by ensuring that shut-down will happen even if one instance erroneously does not request it; 2-out-of-3 (voted) redundancy ensures both that shutdown is requested when needed, and is not requesting when not needed, even if one instance erroneously requests the wrong behaviour. But compared to the 1-out-of-2 design it will be less effective from the former viewpoint, and more from the latter viewpoint⁴;

- for a continuous control function, whose correct functioning is safety-critical, 2-out-of-3 or 2-out-of-4 voted redundancy increase the likelihood of the control system functioning correctly

The choice of architecture (both number of replicas and adjudication function) determines trade-offs between the probabilities of alternative undesired behaviours, for instance reducing the probability of unsafe failure at the cost of increasing probabilities of safe failures.

The kind of diversity employed also affects the degree of improvement offered by a redundant configuration. Typically applying more measures to increase diversity is expected to make the redundancy more effective. There is extensive research on the actual effectiveness of diversity and on how to assess it ([61][66]; www.csr.city.ac.uk/diversity/).

2.19.4 Effects on security

If the purpose of redundancy is to increase the reliability of a security control, then the same considerations apply as in the previous section. Considering the effect on security of redundancy introduced for safety has two aspects: (1) effects of malicious attacks on the safety functions; (2) effects of malicious attack or accidental fault on any security requirements for the redundant component/function (e.g. confidentiality). Introducing redundancy/diversity for the sake of security also

⁴ Because it requires 2 correct requests for shutdown (less likely than 1 correct request) to produce shutdown when needed, but also requires two erroneous requests (less likely than 1 erroneous request) to produce shutdown when not needed.
has more than one aspect: it may focus on improving a security-specific function against attacks (e.g., using diverse intrusion detection engines to make the intrusion detection function more sensitive, or more specific) or on improving the systems resilience against effects of attack ((e.g., by replicating a component than performs an important safety function), typically also improving it against accidental disruption.

### 2.19.5 Trade-offs or synergies between safety and security

Some examples of trade-offs and synergies are given below. The general pattern applies that increasing the number of replicas increases the chances that any given number of them will fail, but also allows a more stringent adjudication algorithm that prevents some unwanted system behaviour (system failure) even if a subset does fail. These two factors can interact in various ways depending on which failure cause (accidental or malicious) and which unwanted system behaviour is considered.

<table>
<thead>
<tr>
<th>EXAMPLE</th>
<th>Variable and trade-offs</th>
<th>Safety</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trade-offs</td>
<td>Increasing number of diverse replicas</td>
<td>improves reliability of a safety function</td>
<td>But may increase the options open to an attacker for penetrating the system, and thus may undermine system or data integrity and/or confidentiality</td>
</tr>
<tr>
<td></td>
<td>Introducing/increasing redundancy</td>
<td>improves possibilities of detecting failures of individual replicas of the function and thus repair/correct them to maintain redundancy level in the long term</td>
<td>.. but may make it easier for an attacker to cause spurious invocations of this mechanism, thus causing denial of service</td>
</tr>
<tr>
<td></td>
<td>Introducing/increasing redundancy in a security control</td>
<td>may reduce reliability/availability of a safety function</td>
<td>.. while reducing the risk of a successful attack</td>
</tr>
<tr>
<td>Synergies</td>
<td>Introducing redundancy/diversity</td>
<td>reduces likelihood of certain unwanted behaviours happening due to failure of redundant replicas due to accidental causes</td>
<td>... as well as due to the same failures if caused by malicious attacks</td>
</tr>
</tbody>
</table>
2.19.6 Interfaces

- the interfaces to a redundant / diverse component (the inputs it receives and the outputs they output) can be the same as for a non-redundant implementation of the same function. Redundancy/diversity are then hidden to the outside world; the redundant component is a plug-in replacement for a non-redundant implementation of the same function.

- however the input signals may be replicated outside the replicated component, to avoid the input interface being a single point of failure. They can be exact replicas of the same signal, approximate replicas (e.g. coming from redundant sensors with loosely synchronised reading by the redundant replicas), intentionally different encodings of the same information, or different signals (e.g. a safety protection function being implemented by diverse replicas of a component, which read different physical variables from the state of the protected plant).

- likewise, the outputs can be replicas of the same signals, possibly to be “adjudicated” (voted or otherwise aggregated); they may be different signals (e.g. actuating different shutdown mechanisms).

- the adjudication function may be performed by components separate from the redundant / diverse replicas of implementation of the function (including voting actuators or diverse actuators, so that the actual adjudication is performed by the physical system controlled), or integrated in them.

- other interfaces may include configuration / maintenance interfaces, which may be common to the various replicas of the component/function, or separate for each one of them.

2.19.7 Matching the kind of diversity to the threats

- Diversity is applied to reduce the risk that the same cause may cause concurrent (common) failures in more than one redundant replicas of a function. The type of diversity required therefore depends on the causes assumed for common failures.

- to protect against errors in design, it is typically required that the replicas be of different designs. This can be achieved by simply procuring two or more implementations of the same specification; but in practice it is common to “force” diversity, that is, require differences, in the design or in the development process, such as to reduce the probability of similar development errors affecting both designs. Errors are possible at all level in the “hierarchy” of development steps and artefacts - from the engineering analysis of the complete controlled system to hardware/software requirements down to machine code, and typically, the higher the level of the errors we wish to defend against the higher the level at which diversity has to be introduced. Thus, for instance, concerns that a compiler fault may introduce faults in machine code, starting from correct source code, can be addressed by using diverse compilers; while a concern about misunderstanding of specifications may lead to producing equivalent specifications expressed in different formalisms. “Functional”
diversity, in which the same system-level objective is achieved through diverse implementations that use diverse technologies and rely on different sensor inputs, actuators and even diverse physical principles\(^5\) is considered to have the broadest coverage of the various causes of common failure\(^6\).

- However, “data diversity” sometimes tolerates common design faults to some extent: for instance, in some complex systems (like server clusters), it has been observed that component failures due to design faults are often tolerated by the use of redundant \textit{identical copies} of software subsystems. This was because their complexity and loose coupling made it unlikely that the identical copies would be subjected to exactly the same demand while in the same state [54]: in these systems, when two replicas process the same input, they are really effectively processing different data points in the demand space (the set of all \{input, internal state\} pairs). Such “data diversity” is often present in embedded systems thanks to having redundant sensors with slight differences in their reading errors and timing (indeed, its presence must be taken into account in designing the adjudication function, lest it cause spurious alarms). If the design faults have a small footprint in the demand space, data diversity may afford substantial reliability improvement. However, experimental evidence is limited.

- At the other extreme of the range of failure causes, diversity of design may not isolate against other causes of common failures, e.g. common high temperatures, EMI or flooding: here, to reduce the likelihood of common failure one needs more effective physical separation.

- So, assessing the value of diversity or other precautions requires relating them to the likely causes of common failures. For software diversity, this is done systematically e.g. in [59] or rules can be set to indicate the desirable ways of seeking diversity in view of an assumed set of risks [69].

\(^5\) As an example, protection (emergency shutdown) for a nuclear reactor can be achieved by two different protection “trains”, one relying on monitoring neutron flux and one on monitoring temperature. As another example, navigation aids on vessels and airplanes usually exhibit functional diversity; etc.

\(^6\) As greater diversity is sought to reduce the likelihood of common design errors, it becomes clear that the boundaries between “modular” redundancy and other forms, like safety monitors or watchdogs, are fuzzy. For instance, two replicas of a control function can be used that have effectively the same function but of which one is only used as to check the correctness of the other, not to drive the controlled system (\textit{cf} Airbus A320 architecture); or at the other end of the spectrum, a control channel may be complemented by a safety-checker channel which however does have authority on the controlled system (\textit{cf} ATC). This difficulty of classification need not concern us in SESAMO. Probabilistic modelling and statistical analysis approaches work reasonably well near these boundaries [75], [67] and extensions can be studied according to needs.
2.20 OVERVIEW OF BUILDING BLOCKS

In this section we give an overview of the building blocks and attempt preliminary analysis of safety and security as well as their cross-influences.

The quantification of safety and security for a building block is made via rating from “+++” strong positive impact, “o” no effect, to “---” strong negative impact to safety and security respectively.

The cross-influences between safety and security are defined via synergies and trade-offs, which are illustrated on Figure 2-5.

![Figure 2-5: Synergies and trade-offs](image)

The quantification of cross-influence is made “+++” lot of synergies, “---” lot of trade-offs required. For example “+++/o” means a lot synergies and no trade-offs required, “+/--” there are some synergies but with some necessary trade-offs, and “o/o” no cross-influence.

Table 2-6 below gives an overview of the building blocks addressed in Section 2 with first attempt for quantitative evaluation of the safety, security, and cross influences as described above.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Safety</th>
<th>Security</th>
<th>Cross-influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Encryption and decryption</td>
<td>+</td>
<td>+++</td>
<td>+++/--</td>
</tr>
<tr>
<td>2</td>
<td>Signature generation and verification</td>
<td>+</td>
<td>+++</td>
<td>o/--</td>
</tr>
<tr>
<td>3</td>
<td>Node authentication</td>
<td>++</td>
<td>+++</td>
<td>++/--</td>
</tr>
<tr>
<td>4</td>
<td>Access control and traffic filtering</td>
<td>+</td>
<td>+++</td>
<td>+/-</td>
</tr>
</tbody>
</table>

**Figure 2-6: Attributes of building blocks**
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Integrity protection</td>
<td>+</td>
<td>+++</td>
<td>++/--</td>
</tr>
<tr>
<td>6</td>
<td>Checksums</td>
<td>++</td>
<td>+++</td>
<td>++/-</td>
</tr>
<tr>
<td>7</td>
<td>Bootchecks</td>
<td>+++</td>
<td>+++</td>
<td>++/-</td>
</tr>
<tr>
<td>8</td>
<td>Software configuration checks</td>
<td>+++</td>
<td>+++</td>
<td>++/-</td>
</tr>
<tr>
<td>9</td>
<td>Run-Time Monitoring</td>
<td>+</td>
<td>++</td>
<td>++/--</td>
</tr>
<tr>
<td>10</td>
<td>Plausibility checks</td>
<td>+++</td>
<td>+</td>
<td>++/--</td>
</tr>
<tr>
<td>11</td>
<td>Logging</td>
<td>+</td>
<td>+++</td>
<td>++/--</td>
</tr>
<tr>
<td>12</td>
<td>Security Audit</td>
<td>++</td>
<td>+++</td>
<td>++/-</td>
</tr>
<tr>
<td>13</td>
<td>Levels of Operation</td>
<td>++</td>
<td>++</td>
<td>++/-</td>
</tr>
<tr>
<td>14</td>
<td>Information Flow Control</td>
<td>-</td>
<td>+++</td>
<td>+/-</td>
</tr>
<tr>
<td>15</td>
<td>Partitioning</td>
<td>+++</td>
<td>+++</td>
<td>+++/-</td>
</tr>
<tr>
<td>16</td>
<td>Virtualisation</td>
<td>+++</td>
<td>+++</td>
<td>+++/-</td>
</tr>
<tr>
<td>17</td>
<td>Protocols for secure real-time communications</td>
<td>+</td>
<td>++</td>
<td>+/-</td>
</tr>
<tr>
<td>18</td>
<td>Redundancy and diversity</td>
<td>+++</td>
<td>depends on architecture and threats</td>
<td>depends on architecture and threats</td>
</tr>
</tbody>
</table>

**Table 2-6 Overview of building blocks**
3 Behavioral Models for Selected Building Blocks

The goal of this section is to show how behaviour of BB described defined in Section 2 can be modelled within SESAMO domains and/or considered tools/tool-chains. Results of this section will give the first crystallisation point for SESAMO tool-chains as well as examples on domain specific applications of BBs within considered tools/tool-chains.

3.1 Modelling of Partitioning in IMACT

The goal of this section is to present a top down modelling approach (or vertical cut of the IMA modelling) for the building block “Partitioning” starting from the top level system design down to the operating system running on specific hardware. To align description closer to real-life, we use the modelling approach in “Integrated Modular Avionics Configuration Tool” (IMACT) being developed at SYSGO as example.

The concept of the IMA is described in the SESAMO Deliverable 1.1. The focus of this section is on the modelling of the building block “Partitioning” in the scope of IMA; more exactly we present modelling of “Partitioning” at the level of Integrated Platform Module (IPM) and its core software. These two terms are defined further in the section.

3.1.1 IMACT Overview

IMACT helps system designer to configure the IMA system. The modelling and configuration scope of IMACT consists of

- **AFDX Network** – AFDX busses, AFDX switches, AFDX Interfaces (including controllers, IPMs, RDCs)
- **AFDX Communication** – Signals, Parameters, FDS, Messages, Segregation and Separation Rules
- **Equipment** – RDCs, AFDX Switches, Cabinets, IPMs, Segregation and Separation Rules
- **Peripheral Busses** – CAN, A429, Signal-Based (AIO, DIO)
- **Power Busses**

IMACT is able to generate configuration for all components of the modelled IMA system, certification artefacts according DO178C/DO254 standards, as well as apply different analysis (e.g. consistency, reliability etc.).

3.1.2 Main components in IMACT

Figure 3-1 shows the IMA architecture that builds the base of the IMACT modelling and configuration framework. The main components are

- **Core software** – The operating system and support software that manage resources to provide an environment in which applications can execute. Core software is a necessary component of a platform and is typically comprised of one or more modules.
- **IPM** – Integrated Platform Module is a module or group of modules, including core software that manages resources in a manner sufficient to support at least one application. IMA hardware resources and core software are designed and managed in a way to provide computational, communication, and interface capabilities for hosting at least one application.
Platforms, by themselves, do not provide any aircraft functionality. The platform establishes a computing environment, support services, and platform-related capabilities, such as health monitoring and fault management. The IMA platform may be accepted independently of hosted applications.

**Figure 3-1: IMA Architecture in IMACT**

- **AFDX end system** – An end node in the AFDX network that can be both sender and receiver of data.
- **AFDX switch** – network-ware responsible for routing AFDX traffic
- **Remote data concentrator** – acts as remote I/O interface

### 3.1.3 Partitioning in IMACT

There are many places in IMACT where partitioning mechanisms are used. The two main elements for partitioning are AFDX network and Integrated Platform Modules. Below we only present their main characteristics from partitioning point of view:

- **AFDX network** – IMACT allows the configuration of AFDX switches (e.g. AFDX subscriber through space partitioning), remote data concentrator (e.g. I/O allocation through space partitioning).
- **Integrated Platform Module (IPM)** –
D2.1 Specification of Safety and Security Mechanisms

- Application Partitions – space partitioning is used to create separated/isolated partitions to run on the same platform hardware
- Partition Scheduling – time partitioning is used to separate/isolate partitions in time running on the same platform hardware
- Resource Allocation – space and time partitioning is used to allocate exclusive or TDMA to platform resources

3.1.4 Building Block “Partitioning” to configure IPM partition

In this section we present how a building block “Partitioning” is used to configure a partition in IPM in IMACT. Figure 3-2 shows a snippet of a model for IPM partition (in XML format) to illustrate usage of space and time partitioning. Figure 3-3 presents an example of IPM configurations: in this example building block Partitioning is used two times, once at application level (see ATA 38 sub-tree) and once at platform level (see ATA 42 sub-tree).

![Figure 3-2: Relevant configuration elements for the IPM partition](image1)

![Figure 3-3: Example of the IPM configuration with Partitioning on Application level (see ATA38) and Platform level (see ATA 42)](image2)

The following considerations apply to this configuration:
D2.1 Specification of Safety and Security Mechanisms

- The **Integrity and Reliability** elements define the maximum rate of losses (“Failure Rate”) or Undetected Erroneous Failures (“Minimum Integrity”) expressed in events per flight hour.
- The **Partition Scheduler (Time Separation)** elements define the scheduling requirements. Each requirement is defined by a period (“timePeriod”) and an amount of CPU to be consumed within this period (“timeBudget”).
- The **IPM Memory Requirements (Space Separation)** define the different memory regions to be used by the partition, as well as certain details, like the access type (“access”). Usually, this building block requires a MMU from the underlying hardware.
- The **Partition Health Monitor** (Partition HM) requirements define the sanctions to errors during each system state. Note that these requirements only include sanctions at Partition and Process level.

### 3.1.5 Generation “IPM Partition” configuration for runtime system

For the sake of simplicity we consider a possible IPM consisting of Freescale P4080 board and PikeOS operating system. PikeOS is a real-time virtualisation platform compliant to the ARINC-653 standard.

Figure 3-5 presents an instance of PikeOS running on top of Freescale P4080. IMACT will generate the configuration for PikeOS including configuration of all partitions with corresponding time and space partitioning. Time partitioning includes schedulers for all partitions running on this IPM. Space partitioning includes allocation of hardware resources to partitions as well as communication ports according to the IMA network configuration.

### 3.1.6 Summary

In this section presented a vertical cut of a modelling approach for the building block “Partitioning”. We showed how standards requirements and concepts (DO178B, ARINC-653, IMA) related to the building block “Partitioning” are modelled in the IMACT tool. We also present how this building block can be implemented on the real hardware with suitable operating system. In the shown implementation example the partitioning on the system level is finally implemented via PikeOS functionality.
3.2 MODELLING OF “PLAUSIBILITY CHECKS” BUILDING BLOCK

3.2.1 Introduction
Plausibility checks are widely used to ensure functional safety of embedded control systems. They can be applied at various levels of the system design which include:

- software design
- configuration/calibration of the system

Both of these application levels for plausibility checks play an important role in the realization of the SESAMO use case “eMotor driver”. First, the eMotor driver is a software component to be realized as a complex device driver in the AUTOSAR architecture. Second, the driver is intended as a generic component, a Safety Element out of Context (SEooC). Due to its general applicability to multiple usage scenarios, the safety mechanisms and part of the behaviour are configurable.

In ISO 26262 part 6 a number of safety mechanisms that are to be applied for error detection on the software architecture level are listed as to be seen in Figure 3-6:

![Figure 3-6: Requirement for plausibility checks in ISO 26262 Part 6 (Software)]

The introduction of plausibility checks into the software are required for any ASIL software component and are recommended for ASIL A – ASIL C. Furthermore, if we consider range checks of input and output data as a special case of plausibility checks, they are strongly required for all ASIL levels. Since the eMotor driver as a generic component will be developed to comply to ASIL D requirements, plausibility checks are mandatory anyway. Furthermore, due to the widely context free realization of the eMotor driver, this implies also the requirement for a flexible and generic model of plausibility check building block.

The second application of plausibility checks are the check of calibration/configuration data. As to be seen in Figure 3-7, this is a requirement for all ASIL levels.
In SESAMO, we introduce the generic model of the plausibility check building block via a Systems Modeling Language (SysML) model.

### 3.2.2 The SysML language

The SysML specification defines a general-purpose modeling language for systems engineering. It is intended to unify the diverse modeling languages used by systems engineers and supports the specification, analysis, design, verification, and validation of systems like hardware, software, information, processes, personnel, and facilities.

SysML has been developed based on the UML for Systems Engineering Request for Proposal (UML for SE RFP; OMG document ad/2003-03-41) issued by the Object Management Group (OMG). The language is based on the Unified Modeling Language (UML) version 2, i.e. SysML extends a subset of UML by several UML extensions mechanisms:

- Stereotypes
- Diagram extensions
- Model libraries

The goal is to reduce the software-centric view of UML towards a system centric view and to provide a language which is easy to learn and use.

Several UML concepts are directly referenced by SysML without extension. These are Interactions, State Machines, Use Cases, and Profiles. Figure 3-8 illustrates the structure of the extension packages. Beneath extensions for modelling of composite structures, parametric modelling, item flow definition, activity modelling and the specification of allocations and requirements, the Model Elements package extends the UML kernel and introduces concepts like views and viewpoints.

---

**Figure 3-7: Requirement for plausibility checks for calibration data in ISO 26262 Part 6 (Software)**

<table>
<thead>
<tr>
<th>Method</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Plausibility checks on calibration data</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1b Redundant storage of calibration data</td>
<td>++ ++ ++ ++</td>
</tr>
<tr>
<td>1c Error detecting codes</td>
<td>++ ++ ++ ++</td>
</tr>
</tbody>
</table>

*a Error detecting codes may also be implemented in the hardware in accordance with ISO 26262-6.*
For modelling of building blocks in SESAMO, especially the plausibility check building block, the block concept of SysML (“Blocks”) is appropriate. A block is derived from the class concept of Unified Modeling Language (UML). It represents a modular unit of a system description and may include both structural and behaviour features. Blocks may have an internal structure. For connecting blocks with its environment, the concept of port is used. A port specifies the services that a block offers to or expects from its environment. As a special case, a flow port may be used to specify data that is entering or leaving a block to be processed by the blocks internal behaviour.

To specify the behaviour, SysML offers the capabilities that are known from UML like activities, state machines, interaction and actions. Which of these behaviour description possibilities is appropriate depends on the context and the language contains these different possibilities by intention.

Furthermore, in SysML it is possible to allocate behaviour to a structural element like a block. That offers the possibility to separate a structural description (using a block diagram) from a behavioural description (e.g. a state machine) and associate both by an allocation at a later point in time. This concept is useful in SESAMO for the plausibility check building block because:

- it is possible to describe the plausibility check as a generic component without having knowledge on the concrete check
- describe the concrete check by any of the behaviour description formalisms of SysML when the context is known and
- bring both together by an allocation

SysML as a UML specialization is also an object oriented language and as such supports the concepts of refinement and generalization. This is applicable to define generic components that are reusable in multiple contexts. For details on SysML please refer to [26] and for details on UML refer to [27].
3.2.3  **Generic model of plausibility checks**

Plausibility checks are widely used in the area for the design of safety related systems. Although it is a generic mechanism, the details of the check depend on the context in which the check is applied. Nevertheless, in SESAMO we have introduced a generic model of plausibility checks as a block described in SysML. This block is expected to be integrated into the different contexts in which it is used across the system. Usually, plausibility checks are used at multiple different locations within a system design. In order to specify a concrete usage of a plausibility check, the general model is taken and specialized using SysML inheritance concept which is available as defined also in other object oriented languages.

As to be seen in, we modelled a block “Generic Plausibility Check”. This block has ports for embedding it in the environment. The incoming ports are:

- “check interval” to specify in which time interval the plausibility check has to be applied
- “signals to check” to specify all input data that is used to apply the plausibility check.

The outgoing ports are:

- “output signals” to specify any output signals of the plausibility check. This could include the original values of the input signals as well as interpolated values.
- “plausible” is a Boolean output that indicates if the plausibility check is successful or not.

The internal structure is given by two contained blocks which are:

- “check behavior” which contains the logic of the check and
- “interpolation” which contains logic for interpolation in the fault case if available.

![Generic Plausibility Check](image)

**Figure 3-9: SysML model of plausibility check as generic block**

The behaviour of the check has to be specified in an appropriate formalism including:
• State machine
• Activity model
• Proprietary technology (e.g. Simulink)

The behaviour model can be allocated to the block “check behavior”. Typically, this is done in a specialization of the plausibility check block in a concrete application context.

### 3.2.4 Cross-influence modelling

The analysis of cross-influence depends on the concrete behaviour model for the actual plausibility check and the various properties as described in 3.2.5. Critical to a plausibility check is the integrity of the input signals. Otherwise, wrong results can be produced and unnecessary or inappropriate actions might be a consequence. Furthermore, plausibility checks require a certain amount of system resources (memory/time) that could have cross-influence with other requirements (like timing) or with other building blocks applied in parallel.

### 3.2.5 Parameters of the behavioural model

This section will describe some of the model parameters that could be utilized for cross-influence analysis described in the previous section. Further set of parameter is not exclusive and can be extended in particular cases. Following parameters of the model are observed:

• Data types that are input to the check
• Rules/behaviour description of the concrete check
• Memory consumption of the software that executes the check
• Run-time for one cycle of the check
• Time interval in which the check is performed

It is important to understand that some of the parameters are more relevant at runtime and others at design/configuration time. E.g. configuration is usually not time or memory critical. However, resource and time related properties should be investigated already at design time.

### 3.3 Modelling of Real-time Communication & Security Building Block

Real-time communication is part of a system that is designed to operate under real-time constraints. System response times are to a great extent influenced by the deployed communication technologies, usually with stringent requirements to temporal bounds and variance. Regarding the deadline for a system response, hard (e.g., control loop) or soft real-time systems (e.g., voice communication) can be distinguished. The following section describes examples of the behavioural model with focus on wireless communication, mixed criticality applications, and safety-critical systems.

The security aspects concern legitimate data traffic, data integrity and sender/receiver authentication.
With the aim of developing a common understanding of both communication aspects and applied security measures, the model should provide a parameterization space which accommodates both safety and security requirements, and provides metrics applicable for either of them, respectively.

Starting from the general description of a behavioural model for this building block, one can take the parameterization list to refine the model to evaluate a specific use case.

### 3.3.1 Cross-influence modelling

From a classification of the parameter space, the main influence (and therefore most important set of parameters) is considered to be parameters characterizing delay behaviour. It is followed by bandwidth considerations for small but regularly sent messages as occurs in safety-critical systems for publishing state. Here, the payload of usually a few bytes is complemented by security information, e.g., authentication data.

If the application under study is designed fail-safe, security mechanisms will contribute to availability concerns, otherwise can affect the probability the system issues a critical event.

One important modelling aspect is the worst-case behaviour of rare events involving both a safety and security relevant event (e.g., caused by one of the modules and propagated within partitions). The model should provide upper bounds of bandwidth and time delays, and if resource allocation on the communication link exceeds available resources or deadlines can not be met, decisions on altered behaviour of communication.

In the event of a security breach or, a reactive approach of the security module can lead to increased information exchange, i.e., higher system load and communication demands and therefore higher channel saturation. Similarly, adaptation algorithms of real-time protocols (e.g., trying to maintain a quality of service level) might mistake rejected/prohibited traffic of security modules and end up generating more traffic of that kind to overcome the observed limitations.

Cross-influence will be observed in mixed criticality application scenarios, because of the different levels of required Quality of Service (e.g., maintaining jitter boundaries), real-time constraints, security demands, and behaviour in case of violated assumptions or degraded communication.

### 3.3.2 Parameters of the behavioural model

This section describes a proposal of main model parameters for the behavioural model, accommodating requirements and metrics for a cross-influence analysis of safety and security aspects.

**Communication**

- Traffic model (per application type per node, or a total of one node)
  - Definition of message types
  - Message sizes
    - Impact on fragmentation, probability of packet errors
  - Probability of message delivery
  - Interarrival time
  - Parameters for schedulability
D2.1 Specification of Safety and Security Mechanisms

- Bandwidth requirements
  - Derived from traffic model plus technology and protocol-specific structures
  - Includes retransmissions of packets, duplications, etc.
- Delay characterization
  - Communication technology/protocol (e.g., CSMA back-off procedures)
- Loss characterization
  - Applying established models for environment, technology, distance, cross-traffic
  - Based on, or extrapolated from, measurements on site
- Jitter characterization
- Technology
  - IEEE 802.11 settings
- (Accepted) failure rates

Security module (Integrity checking, Authentication)

- Rejection rate
- Processing delay
- Additional required data sizes, e.g. authentication tag, key lengths, padding
- Message exchange sequences to establish and maintain operational state of security mechanism
- Parameters to quantify attack behaviour, Mean time to attack, between attacks, mean time to re-establish secure state

3.3.3 Example: Speech (VoIP)

One modelling approach would be to jointly evaluate a) security mechanisms and b) interrelation with other mixed criticality application sharing the same communication link for integrated VoIP applications. The assessment could take report metrics of RTP Control Protocol Extended Reports (RTCP XR) – RFC 3611 – for standard VoIP applications and evaluate those in the described scenario under security and mixed-criticality constraints. For example, evaluation could target operating together with another node-local subsystems having hard deadlines, or tunnelling VoIP packets through a shared authentication module. RTP reports support several types containing statistics, receipt times, duplicate and loss, among other information. The model results obtained should aid in quantification of cross-influence for safety and security, and applicability of certain already established mechanisms, e.g., a mechanism called “thinning”, applied to avoid consumption of disproportionate bandwidth.

3.4 Modelling of Access Controls/Traffic Filtering Building Block

In this section a general model of the “Traffic filtering” building block will be described. Filtering network traffic is one of the most essential security measures allowing access to only legitimate traffic necessary for the proper functioning of a system. The goal of this section is to provide a general behavioural model of the “Traffic filtering” building block that could be easily parameterized and accommodated to various use cases described in the SESAMO Deliverable 1.2.
The general model is shown on Figure 1. It consists of a set of subjects (i.e. devices, processes, etc.) who are exchanging traffic with a set of objects (i.e. devices, processes, etc.) and a traffic unit through which all traffic is routed.

![Figure 3-10: Model of "Traffic filtering" building block](image)

In this model traffic is filtered from one central unit which comprises of several parts: an incoming interface(s), an incoming buffer (adding the processing delay), a set of incoming rules (adding the processing delay), outgoing interfaces, outgoing buffer and a set of outgoing rules. The model could also be simplified by omitting some parts (e.g. interfaces) or extended by adding certain parts or parameters.

The subjects are modelled with a traffic pattern they’re sending towards the objects, and objects are modelled with a traffic pattern they’re sending to subjects. The buffers are used to model packet loss that could occur due to limited buffer size and malicious attacks aiming at denial of service (DoS). Furthermore, the buffers are introducing a processing delay to the building block together with rules that needs to be processed. Defined traffic-filtering rules determine the manner in which the incoming and outgoing traffic flow will be regulated, and also enforce traffic filtering policy; however, a large number of rules could introduce longer processing delay and not necessarily more security. Definition of the rules for traffic filtering depends on information about the traffic. For instance, in IP traffic messages contain information about subjects (source IP addresses, higher protocol data - TCP details – flags, sequence) and objects (destination IP address, port number, etc.) based on which the rules can be defined.

### 3.4.1 Cross-influence modelling

Trade-off between safety and security can be balanced based on the traffic requirements (delay, jitter, and throughput), filtering rules, and criticality level of subjects and objects. Availability of safety-related traffic in the presence of attacks requires trade-off between different model properties. Prohibition rules which are allowing traffic exchange between a subject and an object could also be a source of vulnerability that can be exploited by attacker; consequently, leading to safety-related issues. In general, different mixture and quantity of the traffic rules could yield in different risk level of security and safety. Depending on the particular use case domain, requirements and criticality of concrete subjects and objects involved in the building block. Furthermore, a longer processing delay that occurs due to traffic and rules processing could have impact on buffer overflow, subsequently leading to loss of traffic. This feature of the model could be used in order to analyse DoS attack on the behavioural model.
3.4.2 Parameters of the behavioural model

This section will describe some of the model parameters that could be utilized for cross-influence analysis described in the previous section. Further set of parameter is not exclusive and can be extended in particular cases. Following parameters of the model are observed:

- Buffer size
- Delay and jitter
- Set of rules - rules that can be applied in normal conditions, when intrusion is detected or when certain limit is reached (e.g. discarding or limiting certain types of traffic)
- Throughput rate (bits/s – packets/s)
- Number of subjects and objects and their criticality level
- Failure rate – impact on the system’s safety (decide whether to disable all traffic or not)

3.5 MODELLING OF MONITORING BUILDING BLOCK

In this section we provide a high-level architectural model of the Monitoring building block and discuss architectural options of where we would envisage the monitor to be deployed. The details of deployment will depend on the system context, and we will provide more details on how the monitors may be deployed for the Avionics and e-health use cases later in this document.

As we stated before, a monitor observes and may intervene in the behaviour of the “monitored system”. The events that the monitor observes will be system specific. The details of what exactly should be monitored will be specified according to the safety and security policies and requirements of the respective monitored system. For example, if two different components of an embedded system send (but do not receive) information from to a higher security/safety level component, then the task of the monitor could be to observe whether this unidirectional flow of information is being respected and raise an alarm for, or prevent, any traffic violating this requirement. This requirement may be specified in both safety and security specifications of the system.

Figure 1 below contains a high level overview of where within a system’s architecture a monitor could be deployed and what kind of system components and communication lines could the monitor observe and possibly intervene on. In the following we explain the different aspects of (online) monitoring and architectural deployment options.

---

7 We can think of other examples where this requirement may be reversed: where we want a higher level component to send information to lower level component, but we do not want the lower level components to “contaminate” the higher level component with information.
First, we define the notations used in Figure 3-11:\(^8\):

- M stands for Monitor. A and B are system components. C is the component within a subsystem which makes decision based on the outputs of the two components A and B. E is another system component that lies in the “Outer Subsystem”. Each of the sub-systems, shown with dashed boxes, may run in their own partitions (and hence have their own CPU and memory unit, either directly in hardware or provided via a layer of virtualisation from an underlying operating system (e.g. PikeOS)). The monitor component could make the decisions on the inputs it receives directly, could pass its output to a higher level component for decision making (component D), or it could provide an interface for a human decision maker.
- The arrowed lines point to the parts of the architecture that are being monitored:

---

\(^8\) To simplify the figure we have not explicitly shown interface connectors between the different components and subsystems, but we can assume that the components and subsystems interact via well defined interfaces, which will of course be system specific, and these interfaces allow them to receive and serve requests.
The solid arrowed lines indicate that the aspects (components, communications lines) of the systems that are being monitored may also be intervened on (by stopping, modifying, or delayed a system function from happening);

The dashed arrowed lines indicate the aspects of the system that are being monitored and for which alarms are raised when violations of safety or security requirements occur, but where the monitor does not intervene;

- The solid lines are communication links between different components, or different sub-systems through which these components or subsystems exchange information.
- The solid line boxes are the different embedded system components;
- The dashed line boxes are sub-systems. Each of them may run in their own “partitions” and may have one or more components. These subsystems could be nested within the same architecture and there may be several different partitions for a larger overall system, and these partitions may all need to communicate and exchange information with each other.
- The boxed arrows show the output of the Monitor to either a higher level decision making component (though the monitor itself may also have this functionality built in) or a human decision maker.

In the example architecture (Figure 3-11) we placed the monitor in the “outer subsystem” but yet we expect it to monitor components and communication links in the “Inner subsystem”, “outer subsystem” and “external (sub-) system”. For other architectures this may not be applicable, and the monitor may need to be placed completely outside the overall system that it is monitoring. We will discuss an example of this latter case with the Avionics use case later in the chapter.

In terms of the threat model that may be assumed for the monitoring systems, in the general case we assume that the attacker may be able to launch attacks against any of the components and sub-systems in the figure, including the monitor itself. The types of attacks available and the threat actors would of course differ depending on the system context. For this reason, we need to monitor both communication among different components and the behaviours of each component. In particular, a prevention monitor reacts in order to guarantee the satisfaction of requirements. For this reason, it is important to analyse the threat model of the system we want to monitor also with respect to the requirements. This allows not only on the definition and design of the monitor component but also to understand where it should be placed with respect to the components of the monitored system. For instance, in the e-health scenario, as detailed later in this chapter, the monitor works on one hand by supervising the functionality of the system. It is necessary due to the requirements. It may be applied externally as a hardware module or integrated internally. On the other hand, it works for guaranteeing security requirements, with a particular eye to privacy issues due to the fact that, in the e-health scenario sensitive data need to be shared among different systems.

---

9 In the security literature various terminology is used to identify both the attackers (i.e. the individuals, groups and nation states that launch the attacks) and the types of attacks that they may launch (e.g. denial of service, code injection, buffer overflows etc). The extent to which a given system is susceptible to these different types of attack actors and attack types is often referred to as attack space, or attack vector.
3.5.1 Cross-influence modelling

Clearly a monitor could be developed, configured and deployed to observe and possibly intervene on both safety and security related events.

An example could be a monitor that observes the average CPU usage of a particular system component. If we define the “safe envelope” – i.e. the bounds of expected usage of the CPU for performing a certain function in normal operation (e.g. of an aircraft) – then deviations from that envelope (e.g. much higher CPU usage time) may indicate an unsafe operation, possibly triggered by a security related incident (e.g. a successful intrusion of the component which has led to more than one process being executed and hence may lead to the main safety process being “starved” of CPU cycles).

Security is usually defined to include Integrity, Availability and Confidentiality attributes. Thus there is a synergy with the safety requirements of correctness (Integrity) and real-timeliness (Availability). But there may also be conflicts between the safety and security requirements. These conflicts may be due to confidentiality issues that may result from the need to respect the integrity and availability requirements (Safety or Security)\(^{10}\), but may also occur due to conflicting integrity vs availability requirements of safety vs security. An example of the latter which is relevant for monitoring systems may be: a monitor observes that a component is performing more actions than what its safety requirements permit (e.g. in a communication system between components A and B, it is sending message CCC which is legitimate, but also appending to it message XX, which is not allowed). The monitor raises an alarm to notify a higher level decision maker (either another system component or a human administrator) which decides on the best course of action. If the monitor is also asked to intervene (by say allowing the CCC message to go through, but blocking XX) it would have fulfilled the integrity requirement of the system for both safety and security, but may violate the availability requirement since the delay in the message being sent may fall outside of a safety envelope in a given system context.

In summary we expect the following synergies and trade-offs between security and safety for monitoring systems:

- **Synergies:**
  - Both safety and security events can be monitored using an integrated set of rules in the monitor. In some cases the same event may provide information about violation of either (or both) of safety and security requirements (see the CPU usage example above).
  - Integrity requirements are likely to be highly consistent for both safety and security hence same set of rules in the monitor should be applicable for both.

---

\(^{10}\) An example of this is replication and redundancy of components which will benefit integrity and availability, but will harm confidentiality as the attacker needs to compromise just one of the components in order to compromise its confidentiality.
The availability requirements may also be similar, but the safety real-timeliness requirements may be stringent and hence it may take precedence over either security integrity or availability requirements (see trade-offs below for an example of these conflicts).

There are no explicit confidentiality requirements for safety, but confidentiality breaches may lead to safety incidents if the attacker uses the information to either launch a replay attack, or prepare and debug an attack scenario which they may launch at a later time. Hence monitoring, detecting and preventing confidentiality breaches would make the overall system safer.

- **Trade-offs:**
  - The real-timeliness requirements of safety may prevent detailed real-time analysis of security incidents. This real-time analysis may be required if a decision is to be made on whether a security breach has occurred or not (possibly the monitor itself may be compromised). If real-timeliness requirement is respected regardless of security breaches, then a security breach may lead to an even larger safety failure than would have happened if the real-timeliness requirement occurs. But by doing a security informed safety modelling and analysis, we may be able to exploit the time delays between events to fulfil both safety and security requirements. For example, given a choice between a safety failure SF1 occurring in time \( t_1 \) when a real-timeliness requirement is violated, and a safety failure SF2 (of higher impact than SF1) occurring at a later time \( t_2 \) when a security failure occurs, than the monitor may do the following:
    - Respect the real-timeliness requirement and hence avoid SF1;
    - Use the delay between \( t_1 \) and \( t_2 \) to further analyse and the security breach and either:
      - Prevent it from leading to a safety failure SF2 in real-time at no extra cost, or
      - If the security breach cannot be prevented, instruct the system to fail-safe hence reduce availability but avoid SF2.

### 3.5.2 Parameters of the behavioural model

The inputs to the monitor will be the system specific events that need to be monitored. These will depend on the system context. Examples may include:

- CPU usage time by a process;
- Memory access logs and violations;
- Privilege escalation;
- Standard performance measures of components and their communication channels (e.g. throughput rate)
- Etc.
Some systems may also have properties and events which may not be observable by an external monitor. Hence any statement on the monitoring capabilities of that system would be bounded by this observability constraint\textsuperscript{11}.

Based on these inputs and a clearly defined security and safety policy, rules are defined in the monitor on when alarms should be raised and/or interventions should be made.

The outputs of the monitor will then be to raise an alarm (and/or pass on the output to a higher-level component or a human decision maker regarding a violation of policy), or prevent an event from happening (for intervening monitors).

Based on these outputs we can think of several failure modes for the monitor:

- Failure to enumerate all the events of interest (hence missing events that may cause safety or security failures)

- Incorrect/incomplete definition of rules (in case of rule based monitors) or incorrect/incomplete definition of acceptable system state and behaviour (in case of anomaly-based monitors);

- Incorrect interpretation of correctly defined rules;

- Failure to process rules due to insufficient system resources available to the monitor

- Failure to raise an alarm when it should (false negative) for either safety or security incidents

- Raising an alarm when it shouldn’t (false positive) for either safety or security incidents

The performance of the monitor can then be assessed in the same way as for any decision making tool but looking at its sensitivity and specificity with respect to false positives and false negatives. An additional aspect that is of interest in SESAMO is then to see how the sensitivity and specificity of the monitor differs between safety and security incidents, and what can we learn about the synergies and tradeoffs between these incidents.

3.6 MODELLING OF REDUNDANCY AND DIVERSITY

A component using redundancy, with or without diversity, has at its most basic the logical structure in the following figure.

\textsuperscript{11} For further details on observability constraints see for example http://www.pnas.org/content/110/7/2460
Figure 3-12: Logical model of replicated (and possibly diverse) implementation of a function \( C \) with inputs \( x_1, x_2, \ldots \), replacing a non-replicated implementation with \( n \) replicas of it

This models the logical structure of replication (with or without diversity). A real implementation could match this model, by having separate physical components for each variant, or change it, e.g., it could avoid the adjudicator being a single point of failure by replicating it inside the replicated component, e.g. one copy per replica (variant of the function), or outside the replicated component (e.g. using voting actuators). A physical implementation could even implement the variants as diverse software implementations that run in different time slots on the same hardware. The difference between all these implementations is which kinds of faults (both accidental and “intentional faults”: adversary activities) could be common causes of failure for more than one replica.

Assessing a system that includes redundant/diverse components typically has two aspects

1. assessing redundancy: the proper organisation and structuring of a redundant system. On the design side, this requires identifying the components forming a system and their failure modes, and allocating among other components the responsibility for detecting, masking or tolerating these failures. Assessment must verify that the resulting design contains appropriate defences for each component failure mode considered;

2. assessing common failure probability, or diversity between the failure behaviours of the redundant replicas. If they were guaranteed never to fail together, redundancy assessment would be enough: redundancy against a failure mode of a component would ensure that the redundant/diverse component created in its place has zero probability of that failure mode. This diversity between the failure behaviours can be given precise meanings in terms of the degree of correlation between failures, “beta factors” and similar model parameters applied in combinatorial reliability models. On the development side, this is achieved by separation to avoid common-cause and propagated failures, plus diverse processes (people, procedures and practices) are used to create diverse products that are meant to minimise the risk of coincident failures. On the assessment side, assessing the effects of these various defences is the main challenge.
For the purpose of modelling its internal behaviour, which determines whether (and how) failures at component (replica) level affect external behaviour and thus effect on the broader system, the building block is described:

- the number of replicas, and any form of diversity among them
- the function of the adjudicator, e.g. majority voting, quorum (threshold) voting, etc.
- the aspects of the implementation that affect common causes of failure: that is, common causes to be considered; means applied for diversifying these causes, or the replicas’ likelihood of failure given that a cause materialises; and means for separation among replicas to control propagation of failures.

The first two bullet points affect directly the modelling of redundancy; the last one determines both what combinatorial reliability model is appropriate (e.g., if the adjudicator is a single point of failure, it will be logically “in series” reliability-wise, with the redundant replicas); and what educated guesses can be made for the probability of common failure of the replicas.

The last bullet point above has important implications for security modelling. For accidental faults, it requires for instance that the possibility of common power supply disturbances, failures of the adjudicator function that may defy redundancy (e.g., hardware failure of an adjudicator implemented as a single hardware component, or software failure of one that is replicated but without diversity) are considered; if – as is usual – software faults are concern, then the “diversity-seeking decisions” applied in development are spelled out; if failures due to electromagnetic interference are to be considered, the geometry of the redundant system and the shielding of each component be described. For reasoning about security, it requires in particular a specification of which interfaces are available to the assumed adversaries. For instance, it must be defined whether adversaries could interfere with normal inputs to the component; if the component has administration interfaces that are accessible during operation, adversaries might exploit these; or they might have access to a redundant component during offline maintenance; they might have access to; they might apply radiation at higher power densities than the safety standards consider. As can be seen, deciding which interfaces must be part of the model requires a threat model; just as assumptions about the environment of use determine the set of accidental failure causes that needs to be analysed.

3.6.1 Modelling a redundant/diverse function/component from the system viewpoint

From a system-level viewpoint, the essential parameters characterising a redundant and possibly diverse implementation of a function or component are the probabilities of failure with respect to its various requirements. E.g. if the replicated component has a safety protection function, it can be characterised via a probability of failure per demand and a probability of spurious intervention when not required. If the function/component has separate security requirements (e.g. confidentiality of its internal data), the probability of violation of each such requirement will be an additional parameter.

Modelling of the redundant/diverse function/component may aim to estimate such probabilities, or proxies for them. The simplest such proxy is the Boolean attribute: “the redundant component will not exhibit the failure type X, provided not more than k replicas fail concurrently”, treated below.
3.6.1.1 Deterministic modelling: Redundancy-only, or redundancy and diversity

The simplest form of modelling lists the component (or function) failure combinations that are intended to be prevented from causing system failure (or a specific kind, e.g. unsafe failure) and checks that redundancy is present that, if functioning as intended, i.e. without common failures, would indeed prevent the system failure. This form of assessment gives some guidance for design and is sufficient for assuming those specific system failures as unlikely enough to ignore in assessing system-level risk if indeed common failures are sufficiently unlikely.

The approach can be extended to diversity by also checking that diversity is applied wherever common failures would defeat redundancy.

A variety of this approach is to also assume a certain reduction in probability of failure as a result of application of specific forms of diversity. Sometimes this assumes independence of failures [55], sometimes graduated reductions in probability of common failure as a function of the efforts made to ensure diversity [76]. Although none of these assumptions are justified, they form design guidance, useful to the extent that it requires more extensive precautions when the risk is higher.

Concerns with these approaches are that they may give incorrect estimation of risk at system level and also therefore indirectly impair cost-effectiveness of design. In the specific case of modelling safety and security, an extra concern is that the probability of common failure given malicious attack may be very different from that due to accidental threats only (e.g. random demands triggering design faults in software), and thus it is necessary to explicitly study the risk for the two kinds of concerns.

3.6.2 Probabilistic modelling

Probabilistic assessment must take into account that redundancy and diversity do not guarantee failure independence between the replicas: assessing the probabilities of coincident failures requires extra activities beyond the assessment of the individual replicas of the replicated components.

There is a body of research that describes in probabilistic terms the advantages to be expected from diversity in computing systems. This includes a number of experimental studies; City University London has contributed a large part of the probabilistic modelling results (www.csr.city.ac.uk/diversity/). These results include for instance that failure independence cannot be assumed; diversity in safety configurations is guaranteed to decrease risk; for diverse systems in generic k-out-of-n configurations, pessimistic bounds can be stated on the probability of failure of the redundant component/function. In terms of achievement of a desired level of reliability/safety/security, results concern how diversity improves the probability of achieving them, including the probability of avoiding systematic failure altogether; include confirmation of the usefulness of various measures for increasing diversity. Last, various methods for assessment of diverse systems on the basis of operational and other evidence available have been studied.

An important part of the research results is that eliminating common causes of failures does not imply failure independence for failures due to design faults (bugs). A common failure of two replicas is due to the fact that both contain faults; that the “failure sets” of the two replicas (the sets of {input, state} pairs on which they will fail due to these faults) unfortunately overlap; and that during operation one such pair occurs, through the sequence of inputs that they receive. Removing possible systematic causes of common faults does not remove the possibility of such overlaps (if it did so, the probability of common failures due to such faults would be 0). In the end, the cause of common
failure is that specific demand on the system that happen to cause the failure, and the degree of correlation between failures of the replicas depends only on the frequency of these demands [61].

Assuming that diversity guarantees failure independence would be a convenient mathematical simplification, once it is believed that diversity guarantees very low probabilities of common failure: for instance, given two replicas with individual failure probabilities $10^{-3}$ and $10^{-4}$, and a requirement that the common failure probability be under $10^{-5}$, it often makes little difference whether the real probability of common failure is $10^{-5}$ or $10^{-6}$, or $10^{-7}$; or 0, for that matter. However, that the gain is substantial enough to allow this simplification needs to be demonstrated.

Research findings about assessing the probability of common failure belong to two categories: assessment of what reliability improvement can be expected in a certain redundant/diverse architecture, given certain precautions in development, and taking into account the inherent uncertainty; and methods for assessing the actual reliability parameters of a specific implementation. The two kinds of assessment are related but not identical. Assessment of the results that will be obtained, before the system actually exists, is necessary for development decisions but affected by the uncertainties of development; assessing a specific system can take into account these a priori reasonable expectations, but can also use hard data from the verification and validation activities and, crucially, from operational testing. Last, methods for assessing the final product will offer guidance for designs that facilitate this assessment. For instance [64] proposes a promising approach to assessing reliability of an asymmetric design including a primary, complex replica and very simple channel with safety monitoring functions; the possibility of applying this method adds to the attractiveness of such designs, as making it likely not only that high levels of safety will be achieved but also that it will be possible to demonstrate them.

In terms of results to be expected from seeking diversity in design/development (including V&V methods), guidelines tend to encourage “forcing” diversity and there is a natural tendency towards assigning putative values of reliability gain to precautions that are considered worthwhile [76], despite lack of information about their real effectiveness. In terms of supporting preferences between alternative ways of “forcing” diversity, research has developed models in which the faults in the finished product (replicas) are the effect of a stochastic process, representing the variability of the development (including V&V) processes. These are “conceptual” models – their numerical parameters cannot be fully estimated – but they produce inputs for development decisions, indicating preferences between alternative options in view of the reliability that they are likely to deliver. Results include several theorems from probabilistic modelling about the desirability of intentional diversification [57],[71],[74]. These generally show that given a choice between two designs or two development processes for which equivalent levels of reliability are expected (with various formal definitions of this “equivalence”), in developing a two-replica diverse system we should mix the two rather than use the same for both. The more general results also show that in many scenarios, adding an extra “diversity-seeking decision” (e.g., difference between testing methods on top of other differences in development) is going to improve (formally, at least not to reduce) achieved reliability. Given the inevitable variability of the development process, all these results are valid in a statistical sense: they affect the expected value of the reliability of the diverse implementation, or the confidence in achieving a certain lower bound on it, etc. The models developed derive statistics of reliability achieved for a redundant and diverse system from those for the individual replicas. A current limit of these models is that most apply to 1-out-of-2 or 1-out-of-N safety systems rather than voted systems; a more general set worst-case results for k-out-of-N safety systems has been developed [68], in view especially of the need to consider effects of diversity in areas where little experience is availa-
ble, viz security. The worst cases for expected pfd in the absence of “forced” diversity shows no gain from diversity, showing that assessment a priori must include arguments from experience about the specific effectiveness of the methods adopted. However, even very pessimistic a priori results can be combined with testing results on the specific system to produce useful predictions.

Empirical results also exist - albeit with all the limits of empirical research in software engineering - that, apart from typically showing improvements between one and two orders of magnitude in the average pfd of a redundant system over the average pfd of the components from which it is built, support some confidence that the patterns of effects predicted as possible by probabilistic models actually occur in practice. In particular, an important factor in the effectiveness of diversity with high quality development processes is that diversity acts as an “insurance policy” against the possibility that a good development process occasionally produces a low reliability product (a product in the “tail” of the probability distribution of reliability resulting from the process): using two or more diverse products from such a high quality process substantially reduces the likelihood that the reliability of the redundant component is also unusually poor [72][66].

In terms of assessment of a specific redundant and diverse system, methods for what we have called redundancy assessment essentially amount to systematically checking that redundancy is present against the failure causes for which it is required, and that common-mode failure causes have been accounted for. For diversity assessment and especially probabilistic assessment of a specific redundant/diverse component or system, methods available from research rely heavily on adding to design information or generally a priori assessments the results of statistical testing on the specific system (e.g., [63]). Bayesian inference allows this combination of evidence, and allows credit to be taken for hard evidence of effectiveness.
4 INITIAL DETAILED ANALYSIS OF SELECTED BUILDING BLOCKS

This section provides preliminary analysis of BBs within the SESAMO use-cases. The goal is that use-case providers apply BB identified in Section 2 and, if applicable, considering behaviour and modelling approach given in Section 3. The results is first feedback on usage of BB from SESAMO domains point of view.

4.1 CONSIDERED BUILDING BLOCKS: AUTHENTICATION AND REAL-TIME COMMUNICATION

4.1.1 Analysis in Railway Use-case

4.1.1.1 Use case related description of problem

The ESSI (Embedded Safety and Security Interface) is a device which provides safe and secure communication between safety related applications through open communication systems. Safety related applications are placed in closed transmission system. The definition of open/closed transmission systems can be found in EN 50159 [32]. From the ESSI functional aspect it is very important that in the closed communication system the risk of unauthorized access is negligible (Negligible is a term of risk acceptance category, which mean that the risk is acceptable without the agreement of the railway duty holders).

According to above description, the ESSI provides following main functionalities (Figure 4-1):

1) Create secure communication channel through open communication interface. The main aim of the secure channel is to protection of application data flows against loss of data integrity and authenticity (in terms of EN 50159 – message corruptions and message masquerade) using cryptographic method. Protection against loss data confidentiality is optional. The possible protection methods are using of hash functions (ISO/IEC 10118-1, 2) [33], [34], message authentication codes (ISO/IEC 9797-1, 2) [20], [22]or digital signatures (ISO/IEC 9796-2, 3) [37], [36].

2) Separate the Closed and the Open Transmission System. Attacks against the ESSI from the open transmission system must not propagate to applications in the Closed Transmission System. The worst case scenario is loss of communication between ESSI partners.

Figure 4-1: Generic ESSI use case
ESSI units always work in pairs. A pair of appropriate ESSI units establishes a session that will transmit data between safety related applications. This session is based on UDP/IP. The ESSI is transparent for safety related systems for which the ESSI provides security channel (In other words, the ESSI adds to the communication stack between two safety related applications a new transparent security layer). The ESSI assembles (Figure 4-2) received bytes/packets from Local Communication Interfaces (Figure 4-2) into packet and add to packet protective code e.g. a MAC (Message Authentication Code) tag and send them through the Open Communication Interface (Figure 4-2) to the partner ESSI. The partner ESSI receives the packet on the Open Communication Interface and verifies the MAC tag (authenticity and integrity validation of received packet), if the check is successful the packet is disassembled and disassembled parts are distributed to the appropriate Local Communication Interfaces.

The ESSI can be decomposed into following high level components (Figure 4-3):

1) **Local Communication Interfaces.** The main function of this block is providing a communication interface for safety related application. This functional block is application specific, as the particular implementation depends on safety related applications for which they provide the communication interface (kind of physical and link layer). But it is possible to define main functions which are common for all kinds of Local Communication Interfaces. These functions are following:
   - Receiving data from safety related application.
• Sending data to Multiplexer/De-multiplexer.
• Reading data from Multiplexer/De-multiplexer.
• Sending data to safety related application.

Considered physical interfaces are following:

• Ethernet.
• CAN.
• RS-232.

2) **Multiplexer/De-multiplexer.**
   This component provides following functions:

• Reading data from Local Communication Interfaces.
• Assembling data received from Local Communication Interfaces.
• Adding protective code item into assembled packets.
• Sending packets to Open Communication Interface.
• Reading packets from Open Communication Interface.
• Generation and verification of MAC or signature.
• Disassembling packets into parts corresponding to individual safety related applications.
• Sending disassembled parts to Local Communication Interfaces.
• Providing finite state automat for establishing/cancellation secure relation with partner ESSI.
• Optionally provide encryption/decryption of packets.

3) **Open Communication Interface.**
   This component provides following functions:

• Reading packets from Multiplexer/De-multiplexer.
• Sending packets to partner ESSI.
• Receiving packets from partner ESSI.
• Sending packets to Multiplexer/De-multiplexer.

4) **Service interface/Monitor.**
   This component provides following functions:

• Providing information about actual state of ESSI for maintenance.
• Provide access to logs for maintenance.
4.1.1.2 Use case requirements to Authentication

The security layer of ESSI provides functions resulting from the EN 50159 standard for safety related communication in transmission systems. The security layer provides mechanisms for the secure transfer for safety related application data involving packet origin authentication and packet integrity protection. Other threats like packet replay or deletion are not considered as these threats are handled by higher layers. This protocol layer is positioned between transport layer and application layer of the ISO/OSI stack.

Principle of this protection is based of adding a message authentication code to the packets. In principle three types of protection mechanisms are being considered:

1) MAC (Message Authentication Code) based on block ciphers as defined in ISO/IEC 9797-1. It is possible to use this method for MAC calculation e.g. CBC-MAC together with TDEA (like as EuroRadio) or the more secure version of CBC-MAC together with AES.
2) HMAC (Keyed-Hash Message Authentication Codes) as defined in ISO/IEC 9797-2. This MAC is using a hash-function to create a MAC. Suitable hash-functions are defined in ISO/IEC 10118 - 3.
3) Digital signatures are defined in ISO/IEC 14888-1 [21].

NOTE: In Figure 4-3 the application data flows are drawn as bidirectional, but in fact data flow between two functional blocks involves two independent unidirectional data flows with opposite directions.
Because of the required resources for generating and verifying digital signatures they are not well suited this use case with real-time requirements.

### 4.1.1.3 Analysis of safety and security cross influences

In line with the use-case description, ESSI channels are established between 2 endpoints, i.e., as pair of ESSI units. This corresponds to unicast message exchange. Creating multicast transmissions in that fashion requires either

- Dedicated transmission between the sender and each receiver, with the appropriate shared secret used
- Altering the packet to hold the protective codes for all receivers, with each receiver checking its corresponding section of the protective code

This form of transmission having multiple receivers will either lead to multiplication of required bandwidth, with possible implications on resource allocation for applications with temporal bounds for safety-critical message exchange. Or to the creation of big message sizes due to the additional authentication data, with possible packet fragmentation in the network.

Security attacks targeting information used by the communication layer below the security layer, e.g., UDP/IP addressing, are not covered.

Another point to consider is the ratio of authentication data to message payload. For safety-critical application, where the common exchange of status or control commands involves only small messages, authentication data will incur a noticeable bandwidth overhead. Such overhead has to be specifically included in the applied traffic models.

From view of performance considerations, real time communication can be split into two groups, for different types of applications: hard-real time and soft real time. For the ESSI use case, an application for voice transmission falls into the category of soft-real time applications (meeting time deadlines most of the time), whereas a remote control application would pose hard real-time requirements.

For Quality of Service aspects with respect to voice communication application, for metrics as proposed for example in RFC3611, if applicable, should be included and assessed to impact of authentication procedures during communication.

The processing overhead for creating authentication tag for a message before its transmission results in additional time delay. The overall temporal bounds while passing messages from the application layer down to the communication interfaces have to be adjusted using either a known upper time bound or setting a conservative estimate. This is particularly true for applications which generate packets with high frequency like encoded voice data.

Therefore, one important aspect of safety and security cross influence is maintaining and (re-) evaluating the predictability of timing limits.

Additional if required input to enable authentication (e.g., certificates) is distributed during operation of the system or offline. Furthermore it needs to be analyzed what communication channels is used for as the ESSI communication or via other means. This involves establishing a shared secret, a definition of validity of that secret over periods of time (session), and negotiating new secrets.
Since the ESSI will add authentication data to all assembled packets, a detailed understanding of ESSI operation during those procedures is necessary.

With each pair of ESSIs having their own shared secret, an attack scenario with a successful attack of a single node would not affect other nodes as the attacker cannot masquerade messages from a different node with the knowledge of the compromised node.

Replay attacks (repetition of messages) are handled by a higher layer as given in the use case description. Typical measures against such threat are to use an agreed progressing counter like current time (synchronized among sender and receiver) or round/sequence numbers as input to the cryptographic function. In the described use-case, dedicated additional prevention methods have to be included for that purpose.

Safety-critical data exchange is commonly enhanced by error-correction, duplicated fields or retransmissions for message resilience. With message authentication, the integrity checks and the detection of altered messages can extend or replace mechanisms purely applied for safety purposes and unintentional modification (data corruption). It cannot, however, be used to correct messages (as possible via error correcting codes).

Message Authentication Codes (MAC)

The two mechanisms to be analyzed further are MACs based on block ciphers and MACs based on dedicated hash-functions.

The commonly used way for generating a MAC using block cipher algorithms is the so called CBC-MAC. The principle of CBC-MAC can be found in Figure 4-4.

As CBC-MAC is based on a block cipher algorithm $E$ like DES or AES it can only operate on blocks of $n$ bit where $n$ is e.g. 64 or 128 bit. If a message does not consist of an integer multiple of $n$ bit padding is required. For building the CBC-MAC a block cipher is operated in CBC mode of operation and the last block of this encryption process is the MAC. A message $M$ is subdivided in $i$ blocks on $n$ bit each. If the last block is shorter than $n$ bits it is padded with ‘0’ bits. However, this most trivial padding method is insecure if the size of the message is not fixed. In this case an exis-
Potential forgery of CBC-MAC is possible. Solutions for preventing this attack include the encryption of the MAC using the block cipher and second key also known as Encrypted MAC (EMAC) or the use of a secure padding method.

Cryptographic hash functions such as those defined in ISO/IEC 10118-3 usually require less computational resources than symmetric block ciphers such as DES when implemented in software. With the development of AES that also provides good performance in software this issue becomes less significant. However, hash-based MACs are still widely used and still provide better performance.

HMAC uses a cryptographic hash-function together with a secret key that is shared between two involved parties. To generate the MAC of a message $M$ using a $n$-bit hash function $H$ the following operation needs to be performed:

$$\text{MAC}_K(M) = H(K \oplus \text{opad}) \ || H(K \oplus \text{ipad}) \ || M)$$

Where $\text{opad}$ is the concatenation of $n/8$ times ‘00110110’, $\text{ipad}$ is the concatenation of $n/8$ times ‘01011100’, $\oplus$ denotes the bitwise Exclusive OR operation, and $\|$ denotes a concatenation.

In order to be able to apply the hash function the message $M$ is an integer multiple of 512 or 1024 bits what is the input block size $n$ of common hash functions. The padding significantly impacts the performance of the MAC algorithm for two reasons:

1.) Padding bits increase the size of a certain message. This can in particular increase the required bandwidth for small messages.
2.) Padding increases the number of operations required to generate a MAC and thus affects message delay.

For security reasons also the padding method proposed by e.g. ISO/IEC 10118-1 [33] includes the message length. Thus depending on the message length one additional operation of the hash function’s round function is required to process the padding bits.

In [38] the authors compare the performance of CBC-MAC-AES and HMAC-SHA-1 for different statistics of IP packet lengths. The average time for MAC calculations in software does not differ significantly between both mechanisms. However, for larger packet sizes HMAC performance improves as the impact of padding decreases.

For cryptographic algorithms an e.g. exhaustive key search attack is always possible. Thus, there is a direct relationship between the level of security and the performance of an algorithm as attackers need a computation of the underlying method to verify the guessed key. Based on the time required to complete an operation in [39] different security levels have been defined:

<table>
<thead>
<tr>
<th>Authentication Methods</th>
<th>Security Level</th>
<th>Time (ms)</th>
<th>Evaluation</th>
<th>Output bitlength $n$</th>
<th>Key length $l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC-MD5</td>
<td>0,3</td>
<td>90</td>
<td>weak authentica-tion</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>HMAC-SHA-1</td>
<td>0,6</td>
<td>148</td>
<td>acceptable authentica-tion</td>
<td>160</td>
<td>512</td>
</tr>
<tr>
<td>CBC-MAC-AES</td>
<td>0,9</td>
<td>163</td>
<td>fair authentication</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
\[ \text{Table 4-1: Security levels of authentication methods} \]

Security levels given in Table 4-1 only describe the relationships in the level of security. They are not absolute numbers and represent a relative metric for the evaluation.

The effort required for an exhaustive key search depends on the length the secret key. For a key with length \( k \) an attacker needs \( 2^{k-1} \) MAC operations to find the key with a high probability. If an attacker is aiming at guessing a correct MAC it would require \( 2^{n-1} \) MAC computations if the bitlength of the output is \( n \).

The security provided by HMAC is generally limited by the birthday forgery attack. For methods like HMAC-SHA-1 where the bit length \( l \) of the internal state equals the output bitlength the attack requires \( 2^{l/2} \) MAC operations.

Message authentication is one of the main tasks of the ESSI. However, stringent timing requirements given by safety critical applications must be met also when an application is using a communication channel secured by the ESSI. In addition the additional overhead imposed by adding the MAC tag and the padding needs to be considered [38].

4.1.1.4 Preliminary results and next steps

The next step is to design a model using simulation tools for networks. The aim is to establish a setup for evaluation of applied authentication mechanisms, induced data traffic, and handover scenarios in open communication system over wireless communication links.

Link measurements for IEEE802.11g exist using a prototype setup in a tunnel environment. A selection of mechanisms for the deployed building blocks for the ESSI will show behavior of mobile nodes (train) in wireless communication environment for the targeted application scenarios. Of specific interest is time delay behavior between the ESSI open communication interfaces to the access points mounted along the railway site. Target applications include not only train control but also speech communication from a train operator to the monitoring & control center.

Using the aforementioned model the impact of different kinds of authentication mechanisms will be analyzed considering the provided level of security, the associated overhead, required computational resources and the resulting processing delay. An important security issue is also the distribution of secret keys to the different ESSIs. This is still an open issue and concepts for appropriate key management will be considered as well.

For attacks and failure modes, an attack and fault tree analysis will be conducted to provide a detailed view on occurrence of incidents and threats in critical system components of the ESSI. Results of that analysis are not available yet.

4.1.2 Analysis in Smart Grid Use-case

4.1.2.1 Use case related description of problem

The Smart Grid use case presents a coupling of cyber and physical system. In such systems cyber-attacks on communication links can cause disruption of applications with strict time requirements and affect the physical world causing potential safety-related problems. One of the applications with
strict time requirements is the “Automated Decoupling Carrier Relay” protection systems which operates inside of the substation and on the Substation-DER link (see Figure 52 of the SESAMO D1.2 deliverable). When a fault on the grid is detected the protection device (e.g. Intelligent Electronic Device – IED) automatically transmits decoupling orders to circuit breakers inside and to DERs connected outside of the substation, so they cease electricity supply.

In order to properly protect electrical components of the substation and the grid, the protection devices rely on a communication network often composed of Ethernet switches. Those communication networks can become a subject of an attack that could modify a configuration of protection devices or insert a fake protection message causing potential problems in the grid.

In scope of the building block following problems regarding the protection system will be analysed:

- Timing-requirements for the protection system and the lower bound for tripping of DERs.
- A relevant communication protocol for the protection devices that will satisfy defined time-requirements.
- Potential cyber-attacks and their impact on the system.
- Possible security solutions balanced with real-time performance that could mitigate the attacks, i.e. authentication and key management.

One example of potential communication protocol is The Generic Object Oriented Substation Events (GOOSE), defined by the IEC 61580 standard that is typically used for message exchange between protection devices. As described in [7], for the GOOSE protocol communication latency is one of the main issues in implementing a security solution.

### 4.1.2.2 Analysis of safety and security cross influence

In general, adding security solution to the protection system will enhance the safety of the whole system since attackers would not be able to insert fabricated messages. For safety and security cross-influence analysis in this use case, it is necessary to simulate both cyber and physical part of the use case, i.e. the electrical grid and the overlaying communication network. Particular interest is to simulate potential attacks on the protection protocol and the communication network and evaluate their effects on safety in following scenarios:

- DER disconnection request is sent during critical peak period, causing power shortage/outage
- Malicious order sent while crew members are on the grid network – working under tension.

For scenario evaluation a sample test-case of the grid containing a substation and arbitrary amount of DERs will be defined. For the test-case of the electrical grid an overlaying communication network will also be defined in order to simulate a security solution and potential adversary attacks on the protection protocol (Figure 4-5).
D2.1 Specification of Safety and Security Mechanisms

Figure 4-5: Abstract view on simulation scenario for communication network

For simulating the electrical and the telecom aspects of the use case appropriate tools will be chosen according to expertise of partners who will conduct the analysis.

4.1.2.3 Preliminary results and next steps

In scope of this deliverable the preliminary result gives a review of security threats on communication network and challenges in implementing security mechanisms in the smart grid based on state-of-the-art. At the end of this section the next steps for analysing the use case are described. The future smart grid is expected to rely on robust communication networks to provide efficient, secure and reliable information exchange. Distributed system operators (DSO) will face new challenges in implementing above mentioned attributes in variety of the smart grid communication networks that will connect new systems such as the DER management, the demand-response or the DER site towards control center (bottom-up) and from control center toward substations (top down).

Nowadays, the DSO controls and monitors a portion of the grid from a central point using SCADA system. Typical SCADA system contains up to a few hundred substations that are geographically widespread. The substation contains measurement devices for voltage, state of circuit breakers, isolators, transformer tap changes, and other IEDs [2]. Measurements are collected by Remote Terminal Units (RTU) placed in substations and sent to the SCADA system through a communication gateway (connected to a backbone WAN) every 2-4 seconds. Operators at the SCADA system monitor the grid for alarms and dispatch new commands to the RTUs. The same way, DERs connected to the grid could be controlled by DSO from the central SCADA system. Information flow in those systems goes in two ways: from substation/DER toward control center (bottom-up) and from control center toward substation (top down).

However, inside of the substation IEDs have to be able to communicate in peer-to-peer manner over local network in order to enable fast response for protection applications. As described before local substation network could also be connected with DERs for protection purposes. Those communication networks could be attacked using various techniques targeting network availability and data integrity. For mitigation of denial-of-service attacks its necessary to first evaluate possible consequences of that attack, thus simulation of cyber-physical system is needed. Countermeasures to attacks and challenges in implementing a security solution are [9]:

i. Authentication protocol and key management to protect integrity of messages exchange on the network strong authentication scheme is required.
The protocol should have small computation cost, low communication overhead and robustness to attacks.

ii. Intrusion detection mechanism should be able to detect intrusion attempt into substation and react accordingly to prevent an intruder to gain unauthorized access.

iii. Firewall design for control and filtering of traffic can be performed according to information flow patterns.

Further results of the building block analysis will be available after the test-case is set-up which understands the following steps to be taken:

- Definition of a full test-case - including both the electrical and the telecom part
- Implementation of the test-case with appropriate simulation tools
- Definition of the protection system with relevant parameters (e.g. timing requirement, communication protocol, message format, etc.)
- Analysing impact of attacks, with focus on safety-related problems defined by the scenarios
- Defining a security solution for the protection system and balancing it with respect to performance constraints

4.1.3 Analysis in automotive (CAN bus)

4.1.3.1 Use case related description of problem

In the eMotor use case, the requests for the motor controller software may come from the other nodes on an automotive network such as CAN. As was mentioned in Section 2.18 CAN bus does not, by default, provide any means for authentication of message senders. Authentication must be added on top of standard CAN bus protocol by software means, e.g. by appending message authentication code (MAC) to some messages. There are several limitations that limit how this can be done: limited size of CAN frame and limited CAN bandwidth.

To find optimal assignment of communication parameters such as size of MAC in bits, message priorities (identifiers) and the frequency of sending authenticated messages, it is necessary to create a model of communication over CAN bus. This model must consider all systems that communicate on the CAN bus. eMotor software will be just one of such systems.

4.1.3.2 Model of CAN bus communication

In order to create a security enhanced model of CAN bus communication we proceed from classical CAN bus schedulability model presented in [41]. In this model, every message $m$ is characterized by the number of data bytes $s_m \ T_{mq}$. Based on the number of data bytes, one can calculate maximum message transmission time $C_m \ T_{mq}$. Queuing of the message for transmission is triggered by an event with minimum inter-arrival time of $T_m \ T_{mq}$. Actual queuing happens at time interval between 0 and $J_m \ T_{mq}$ after the occurrence of the triggering event. The value $J_m \ T_{mq}$ is known as queuing jitter. Each message has a hard deadline $D_m \ D_{mq}$ that denotes the time since the triggering event by which the message must be received by the nodes that require it.
Based on the above outlined model, classical schedulability analysis [41] calculates the worst-case response time $R_m$, i.e. the time from the triggering event occurring to the message being received. If $R_m \leq D_m$ for every message in the system then the system is said to be schedulable.

When message authentication code (MAC) is added to the message, it increases the size of the message and its transmission time. Depending on the protocol used (e.g. [42] or [43]), MAC can be even transmitted in a separate message, which increases response time even further. This clearly has impact on schedulability. Deciding which kind of MAC to use for each message is not easy, because of the inherent trade-off between security (size of MAC) and schedulability (required by safety). In order to help engineers with selecting the mask we propose to extend the CAN schedulability model with the following parameters. Security level $L_m$ which indicates the strength of MAC added to the message. The highest security level means that long, hard to break; MAC (e.g. 128 bits) must be used, which implies using extra messages for sending the MAC. The lowest level means that no message authentication is necessary.

For certain signals (values transmitted by messages), it is not necessary to append MAC to every message, because the physical properties of the signal prevent it from having arbitrary values. For example, engine temperature cannot change from 0 to 100 °C in one second. In this case plausibility checks may be applied to messages without MAC in order to discover forged messages [43].

Therefore, the other parameter of the model is the MAC frequency $f_m$, which is an integer number with the meaning that only one of $f_m$ messages needs to have MAC added.

The last parameter that is needed for the envisioned analysis is the time needed by the recipient for checking the MAC. Only after the MAC is checked, the message can be processed by the application.

### 4.1.3.3 Analysis of safety and security cross influences

The model of CAN bus communication, presented in section 4.1.3.2 will be used to add optimal communication parameters with respect to safety and security. Specifically, we aim to find best possible security parameters such as MAC size and message priorities that allow the system to remain schedulable. This analysis will be delivered as part of work package 3.

### 4.1.3.4 Preliminary results and next steps

The work on the analysis was started recently and there are not any significant results yet. The next steps are clear – finish the analysis and publish a paper with the results.

### 4.1.4 Analysis in medical use case

#### 4.1.4.1 Use case related description of problem

Time-related criticality appears if data is exchanged in any direction. Basically problems occur at the interfaces of the system. In the medical use case’s data concentrator these are:

- 1a: sensor -> core unit: data plausibility check (absolute value, delta check)
- 1b: (sensor + ADC) -> core: data integrity check, i.e. signature
- 2: inter core interface: checksum, integrity by time stamp & ID
- 3: core unit <-> operator: Authentication, data encryption, Message authentication code i.e. AES, access control

In the medical use case the major trade-off is between data reduction and security and safety. According to the architecture (for further information please have a look at D3.1) data reduction and
the security relevant tasks authentication and encryption are executed in different software components and in practice running on different hardware modules. While the operation of the system as a whole is managed by the core component that calls the other software modules the trade-off basically appears between these two modules.

Every task shall be initiated by the core component except incoming data for system re-configuration. Therefore, the components shall send an acknowledge message to the core component at the time they receive the task and a done message, when the task is completed. If the components fail to send these messages in a defined time span, the system shall return to safe state.

The following software components interact with the core components

<table>
<thead>
<tr>
<th>Software component</th>
<th>Security measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication/Encryption</td>
<td>Encryption for outgoing data cannot be detected, unless two independent encryption units are implemented and their output is compared against each other. If possible, only one encryption unit shall be used, if fail-safe operation can be guaranteed (insured by design), otherwise two independent units shall be implemented</td>
</tr>
<tr>
<td>Key storage</td>
<td>The stored encryption key can be verified by a checksum</td>
</tr>
<tr>
<td>System configuration</td>
<td>The system configuration can be verified by a checksum</td>
</tr>
<tr>
<td>Data integrity/Validity</td>
<td>Data integrity verifies the checksum of incoming and outgoing sensor data</td>
</tr>
<tr>
<td>Data storage</td>
<td>No security check needed, done by data integrity component</td>
</tr>
</tbody>
</table>

Table 4-2: Software components and security measures

### 4.1.4.2 Analysis of safety and security cross influences

Data reduction limits the amount of data that is processed and stored within the system and transferred to the doctor. Data reduction is necessary to limit the required bandwidth for the data transfer. The following steps will be taken to achieve the reduction:

- A time stamp will be written into the memory before the first data is recorded. Every data includes a unique ID. Therefore, it is not necessary to store a time stamp for each data set, since data will be acquired in a defined time interval. A new time stamp will be generated if the system has to return to a safe state.

- Data is only stored if its value has changed compared to the previous data. Due to the unique ID, no information will be lost

- In some cases it might be possible to reduce the ADC resolution since it might not be required for certain measurements
Although it is desired to reduce the amount of data, it is necessary to add redundancy to every data set to insure that transmission errors or misread data can be detected. This will be achieved by a cyclic redundancy check (CRC).

From the user’s perspective the trade-off is between time-critical requirements and encryption and authentication. In the medical use case it is not a real-time requirement actually. Criticality over time arises firstly, due to the fact that the data to be collected is medical data and, secondly, due to the fact that a data concentrator needs to supply the data when it is requested. Therefore time-criticality breaks down into two different requirements concerning different modules inside the system and demanding different methodologies to deal with. In terms of time-criticality two requirements should be met:

- Since medical data has to be collected in time, which means the system shall be able to react within an interval of time which is short compared to the time interval the medical data need to appear or change. I.e. if a heartbeat is to be recorded data needs to be recorded as fast as the heartbeats appear. A remedy for this trade-off between the response time and security may be an intelligent method of reducing the data, which means regarding unnecessary information.
- Secondly, the data has to be accessible at the time it is demanded. This affects not only the system’s response time but handling an unreliable data transmission path as well.

Since it is required to transfer the data via Ethernet, two different protocols could be used for transmission, UDP or TCP. While UDP is a fast protocol it only supplies a unreliable transmission path. On the other hand TCP implements a handshake protocol with handling of lost packets. Due to the overhead for the handshake, the effective transmission rate of the use data is lower than for UDP. Since the correct transmission of the data is more important than transmission speed, data will be transferred by TCP.

### 4.1.4.3 Preliminary results and next steps

As described above, the message transferred to the operator should contain:

- The encrypted information, because data is private
- A CRC, for safety reason
- A message authentication code (MAC), in order to protect against modification

An optimization of the transferred data may be implemented since the purpose of the CRC, check for unintended corruption of data can also be fulfilled by the MAC. The differences and consequences between both options must be clearly stated:

- A CRC could be used to correct an error while this is not the case of the MAC.
- The safety related function “CRC check” is replaced by a safety related function “check MAC” on operator side

In the other direction, operator to data concentrator, for the configuration of the system, the removal of CRC is an option including attachment of the safety requirement to the check of the MAC. This means that the module doing encryption in order to check the MAC would be safety relevant.

As for the operator side (i.e. on PC) we would have the requirement to develop a function (including hardware and software) that must fulfill a safety and security standard. On the software this would mean e.g. to be compliant to a safety coding guideline and to a security coding guideline.
The next step will be Software development. After applying the security and safety Mechanisms by implementation of the architecture as proposed in D3.1 an analysis can be performed. Since the tool chain for software development has been set up already and easy-id has been looking into the hardware provided by Infineon, the development of components of the architecture can start now. In order to keep the complexity as low as possible the analysis of security and safety mechanisms has to start bottom-up, which means from the smallest unit of the system, i.e. the software-components. This approach should enable the most concise and less error-prone analysis.

4.1.5 Analysis in Industrial Drives use case

4.1.5.1 Use case related description of problem

The Industrial Drives use case is a typical motion control application out of the field of industrial automation and control. Motion Control products cover a large variety of variable frequency inverters for synchronous and asynchronous motors ranging from standard electric motor systems and servomotors for Motion Control applications including linear and torque motors to motors for use in hazardous explosion areas, to high voltage, DC and customized electric motor systems.

This use case focuses on a generic commercial motion control platform solution for permanent magnetic synchronous motors (PMSM), see Figure 4-6. The typical application is within e.g. tooling machines. Since this application is based on a generic control reference platform it is also possible to target application in similar domains like Electric Drive Trains in automotive domain.

![Figure 4-6: Industrial Drives Demo Installation [Source: ALTERA]](image_url)

The large variety of communication and sensor interfaces of such embedded systems adds significant security challenges to the safety mechanisms already implemented in today's commercial industrial products. Authentication is one of the fundamental underlying principles of secure communication within industrial applications. It is a challenge on the higher IP based communication layers within industrial automation as well as on the lowest sensor communication levels. Within the current use case the authentication aspect is analyzed in four areas, which were identified as critical, depicted in Figure 4-7. Each aspect is then briefly elaborated in the following section.
Node Authentication of the FPGA:

Also in industrial application the threat of counterfeiting has to be considered. Often because of commercial aspects, but also following safety concerns, it is in the interest of the system provider ensure the integrity of the certified system by locking out unknown components. Node authentication is the mechanism that provides a unique identification of components. Based on this, system access can be granted or be denied, respectively whole components can be rejected.

The challenge here is to establish a secret, ideally not vulnerable and not replicable property of a component. The most effective mechanisms for unique component identification within the field of microelectronics are the so called Physically Uncloneable Functions (PUFs). This topic is still in focus of academic research even though there are first commercial products available. Basically a PUF can be seen as a digital fingerprint or digital DNA that
uniquely is related to a piece of hardware. For the given use case the applicability of commercial PUF solutions is assessed and in the end a suitable option is applied [44]. The use case scenario assumes that the controlling application on the remote host periodically verifies the identity of the responding target FPGA for each access.

1. **Authentication in the context of flash-based FPGA configuration update:**
   On the one hand the FPGA configuration bitfile that holds the description of the whole design must be protected against unauthorized readout and modification while on the other hand an update from reliable sources has to be permitted. So the focus within this context is on suitable means to guarantee that the bitfile comes from a trusted source.

2. **Authentication for means of restricted access to the debug facilities of the FPGA:**
   Current processor based SoC offer an extensive debug solution. The industrial drives use case is based on the he ALTERA SoC FPGA which hosts two powerful ARM processor cores which can be accessed via the JTAG debug interface. This debug interfaces are a highly vulnerable entry point to the system where an attacker could get full control over the system. Of course this interface needs proper protection. For this means ARM offers an admittedly weak debug port protection mechanism based on additional interface signals. This mechanism will be assessed and security extensions will be proposed.

3. **Authentication of the encoder messages:**
   Within the given system several sensors provide measurement data which is processed by the subsequent calculation stages. A central role in this scenario plays the encoder interface, where rotation related data is sampled and transmitted via the internal bus system towards the CPU. The transmission of this data occurs at high frequencies to permit accurate motion control of the attached motors. Establishing secure communication based MAC on this interface is a challenge because of the timing constraints related to the real time critical nature of the control loop. In a future scenario it is planned to have the sensor data directly authenticated within the sensor, end to end.

4.1.5.2 **Analysis of safety and security cross influences**

1. **Node Authentication of the FPGA:**
   For the given use case the security benefit gained by node authentication can cross influence system safety because of additional delays added by the authentication mechanism but also because all additional authentication related HW and SW components must comply the target SIL, since they can corrupt the system in case of malfunction.

   Especially the aging characteristics, and error rates of the used PUF has direct influence on system safety since the aging effects for the PUF related elements can be significantly different from the standard components [45].

**Parameters:**
- Characteristics and delay time of PUF based authentication algorithms
- Update rate of the higher level motion control algorithms
Safety Analysis Method:

- Mathematical WCET calculation of maximal update rate

Security Analysis Method:

- Analysis of PUF characteristics like Uniqueness, Reliability, Uniformity, Randomness, Aging

2. Authentication in the context of flash-based FPGA configuration update:
The safety-security cross-influence in this area can be formulated in short as:

“The better the bitfile is protected against manipulation the higher the chance that the system’s safety functions are working as intended.” Modeling and analysis of this aspect would reduce to looking on the general security properties of the authentication methods in place and therefore is not elaborated in this place.

3. Authentication for means of restricted access to the debug facilities of the FPGA:
Very similar to above flash update aspects the protection of the debug facilities is also only indirectly influencing system safety. If an attacker gains unauthorized debug access to the system he can alter all safety related functionality.

This aspect is very important but not suitable for cross-influence analysis.

4. Authentication of the encoder messages:
The control algorithm loop is very timing critical since its maximum update rate directly influences motion accuracy and motor efficiency. More important from safety point of view is the predictable real-time behavior of the system. Within this context the predictable maximum update rate is essential.

Parameters:

- Message Bit length
- Additional MAC Bit length
- Update frequency

Analysis Method:

- RTL Simulation model
4.1.5.3 Preliminary results and next steps

1. Node authentication of the FPGA:

Preliminary results:

   - An initial paper research on concepts and implementation of PUF solutions for FPGAs has been conducted within SESAMO

Next steps:

   - A suitable PUF implementation will be integrated into the use case demonstrator in WP5 to verify the results of the previous theoretical analysis

2. Authentication in the content of flash-based FPGA configuration update:

Preliminary results:

   - According the initial analysis of this aspect there was no relevant safety-security cross-influence identified.

3. Authentication for means of restricted access to the debug facilities of the FPGA:

Preliminary results:

   - According the initial analysis of this aspect there was no relevant safety-security cross-influence identified.

4. Authentication of the encoder messages:

Preliminary results:

   - First rough estimations regarding the timing overhead were already done:

     The update rate of the drive circuits can be quite high. E.g. a motor operating at maximum of 15000 RPM, or 250 revolutions per second with the assumption of eight pole pairs (number electromagnetic windings in the stator) and an 80 times oversampled sinusoidal current waveform results in a sampling rate of 160000 per second, and thus a process latency of 6.25 $\mu$s.

Next steps:

   - define authentication method and mathematically derive resulting MAC overhead
   - calculate theoretical maximum update frequency
   - implement a mechanism and prove mathematical results with system simulation (long term goal!)

4.1.6 Summary of used models

The preliminary analysis of the cross influences between authentication and real-time communication requirements shows that in many of the SESAMO use cases this issue needs to be considered. We have analyzed the different use cases and identified the problems that in most cases require further analysis to come up with a solution.
A variety of models is being used to do the further analyses of the different use cases. For further analysis probabilistic models as well as deterministic models or a combination of both will be used to gain a better understanding of the use case specific relationships of safety and security.

### 4.1.7 Outlook and Generalisation

This initial analysis provides an overview of the problems around authentication and real-time communication that is prevalent in different use cases. Although there many similarities the different use cases have different ways for addressing this issue. In general there are two areas where a trade-off is requires further investigations.

- The integration of security mechanisms into systems where security has not been considered during design phase often poses challenges to the solutions. These limitations can be related to limited storage or bandwidth like in the use case CAN bus and health monitoring. In these cases security needs to be adapted to the limitations provided by the use case. For MAC this means if there is not sufficient space to transfer a complete MAC it needs to be truncated what impacts the security of the MAC as truncation decreases the effort required for a successful attack. As there is no safety without security it is thus also a safety issue.

- Another limitation of embedded systems is the available processing power. For safety reasons different kinds of messages like e.g. control messages or measurements data in control loop have strict timing limitations that need to be met also when security mechanisms are applied. Like described in Section 4.1.1 there is a relationship between the performance of an algorithm and its level of security given that there are no known weaknesses. If the timing of a mechanisms is critical it is in some cases required to decide for the weaker mechanisms that provides better performance but still a sufficient level of security. Thus, there is a clear trade-off between safety and security required in these cases.

As a next step models of the communication and control systems will be developed in order to understand what the impact of adding a specific MAC mechanism. Based on this analysis the trade of between performances of the algorithm, the bandwidth requirements and the level of security can be made. This will result in optimized security mechanisms which fit best the problem and the requirements of the use case.

The approach that is followed by WP2 is mainly that security is being defined in a way that safety requirements are met. However, we will also consider the case where safety needs to be adapted to given security requirements and how it impacts the system.

### 4.2 Monitoring in Embedded Systems

#### 4.2.1 Analysis in use-case avionics

##### 4.2.1.1 Use case related description of problem

The aircraft IT infrastructure is expanding with each aircraft programme; the extension comprises enhancement of features for existing applications or addition of new applications and functionalities. From the architectural point of view these applications are grouped based on their scope of application into several domains. Each domain has an associated security level. Another domain...
characteristic is the networking technology used for intra-domain communication; the inter-domains communications may need specialized equipment translating from one networking technology to another. The network communication choices are also related to the security level needed for a given domain. Communication between aircraft applications, either aboard the aircraft between different domains or between on-board applications and ground ones, should follow security rules protecting each domain according to its security level defined by the architecture.

Without making reference to any existing aircraft architecture, the ARINC 664 Part 5 is defining the concepts of aircraft domains while ARINC 811 facilitates the understanding of aircraft information security (see Figure 4-8 taken from [48]). Airbus keeps these definitions at a high level for its aircraft. For details of the definitions of various domains please refer to [48] and [49].

![Figure 4-8: Aircraft Security Domain according to ARINC 811](image)

Figure 4-9 (taken from [48]) depicts an integrated system with two security domains communicating unidirectionally. The restriction of one direction is for illustrative purpose only and does not restrict our implementation. Each security domain can comprise one or more partitions. The integrated secure gateway, in the centre of the picture, comprises two partitions belonging to different domains.
The gateway design is split into two parts, outbound and inbound. These parts belong to different security domains and are connected via a unidirectional OS-provided communication channel (Figure 4-10, taken from [48]).
4.2.1.2 Analysis of safety and security cross influences

The Monitoring component in the Avionics use case is placed outside the monitored system (i.e. in a different partition); the same is likely to be true in many other application domains. This is because of the stringent partitioning requirements for avionics systems. The monitor observes and logs the events that happen in the gateway and in the components, but does not intervene – i.e. it does not prevent the events from happening. It is up to a different gateway component (or a higher level component than the gateway itself) to then decide on the intervention.

The monitor in this use case can be configured to observe (and if necessary intervene on) both safety and security related events.

Confidentiality is not a concern for this use case. But integrity and availability (for both safety and security) are. The monitor needs to monitor for security intrusions that may lead to safety failures, but at the same time not violate the real-timeliness requirements of the gateway or its sub-components. Because the monitor will reside in a separate partition, we expect the real-timeliness requirement will be respected (as long as the monitor runs in a non-intrusive mode). However, the identification of the security breaches should at least lead the gateway to decide running in a degraded mode.

In summary, we expect the following synergies and trade-offs between security and safety for monitoring systems in the Avionics use case:

- **Synergies:**
  - Both safety and security events in the gateway can be monitored using an integrated set of rules in the monitor. In some cases the same event may provide information about violation of either (or both) of safety and security requirements violations.
  - Integrity requirements are likely to be highly consistent for both safety and security hence same, or highly overlapping, set of rules in the monitor should be applicable for both.
  - The availability requirements may also be similar, but the safety real-timeliness requirements is more stringent. Hence the monitor is running in a separate partition to avoid violating any real-timeliness requirement of the gateway or its components.
  - There are no explicit confidentiality requirements for this use case.

- **Trade-offs:**
  - As we stated before, the real-timeliness requirements of safety may prevent detailed real-time analysis of security incident which may be required if the monitors are to make a decision on whether a security breach has occurred or not (and hence whether a safety failure may then follow as a consequence of this security breach).

4.2.1.3 Preliminary results and next steps

We have done a preliminary analysis of the monitoring building block, by reviewing existing literature, identifying relevant examples of synergies and tradeoffs of security and safety requirements of monitoring systems, and a first attempt at describing how the monitors could be used in the Avionics use case.
We plan to explore further the modeling of the Monitoring building block, by analyzing the cross influences and tradeoffs of security and safety. We are particularly interested in analyzing the models that allow us to study the sensitivity or specificity of the monitoring building block, with regard to either safety or security incidents, and the interplay between the two types of incidents. This would allow better decisions to be made on the configuration of the monitoring systems by providing the decision makers with an analysis method for quantitatively assessing the trade-offs between security and safety requirements.

We will also explore the possibility of obtaining some real data for one of the use cases in SESAMO, preferably the Avionics use case which we have been analyzing so far.

We also plan to do a more thorough analysis of the avionics use case with a more detailed catalog of examples of synergies and trade-offs of security and safety requirements that may result from the use of monitoring in Avionics.

4.2.2 Analysis in Medical Use case

4.2.2.1 Use case related description of problem

The e-health use case mainly focuses on home applications, known as Mobile Ambient Assisted Living (AAL) Systems. Those systems offer great opportunities in the medical sector because they are intended to supervise patients remotely using multi-parameter biosensors and secure communication networks in order to improve the quality of medical care as well as guaranteeing privacy of personal data. In Figure 4-10 there is a graphical representation of basic concepts involved in the use case.

In this scenario, any kind of system failure can affect a patient’s health or even cause death indirectly at worst. If a patient’s monitoring system is “outsourced” to the patient’s home, consequences of a system failure are critical. Given that the AAL-System is located at the patient’s site, i.e. inside a trustworthy environment the major endangering originates from external interfaces for data transmission.

Furthermore, the kind of data that are exchanged in this scenario are sensitive and a leakage of some information can damage, not only the health but also the privacy of the patient. Hence, mechanisms for guaranteeing privacy and confidentiality during the transmission phase are needed.
4.2.2.2 **Analysis of safety and security cross influences**

In this scenario, the monitor work is twofold: It is placed internally (watchdog) and in this case it works in a passive way. Indeed it does not interfere with the behaviour of the system but just monitors that everything proceeds as expected. On the other hand, it works as a prevention monitor, and in this case it is distributed on several component of the system. The goal of the prevention monitor is to guarantee that the communications are performed using secure channels (using cryptographic protocol) and that access rules are enforced.

Confidentiality, integrity, and availability (for both safety and security) are very important aspects in this use case. The monitor needs to monitor for preventing security intrusions without causing safety failures as, for instance, a wrong transmission of patient data to a medical doctor. Another important aspect that in the e-health scenario is critical is the reaction time. Hence the monitor must be efficient in terms of computation time for not delaying the transmission of data from the patient and the doctor and back.

In summary we expect the following synergies and trade-offs between security and safety for monitoring systems in the e-health use case:

- **Synergies:**
  - Both safety and security events can be monitored using an integrated monitor (watchdog).
  - Confidentiality, integrity and availability are fundamental aspects that have to be considered from both security and safety point of view.

- **Trade-offs:**
  - Also in this use case as in the Avionics one, the real-timeliness requirements of safety is very important but may prevent detailed real-time analysis and enforcement of security requirements.

4.2.2.3 **Preliminary results and next steps**

In this first period of the project, we have studied and analysed the literature on prevention monitors with an eye to their application to the e-health use case.

We aim to provide a more detailed description of a prevention monitor model able to guarantee safety and security in particular in the selected use case in which the monitor has to guarantee, confidentiality, integrity, and availability of sensitive data.

4.2.3 **Summary of the used models**

We have done a preliminary analysis of the monitoring building block, by reviewing existing literature, identifying relevant examples of synergies and tradeoffs of security and safety requirements of monitoring systems, and a first attempt at describing how the monitors could be used in the Avionics and e-health use cases.

Comparing the analysis of monitoring in the two use cases, we can already see some synergy with integrity and availability requirements, but also some differences regarding the confidentiality requirements. Confidentiality is not a concern of the monitoring component in the gateway described in the Avionics use case, but is a major concern in the e-health use case.
4.2.4 Outlook and Potential for Generalization

As stated previously, there are several avenues that we wish to pursue for further work and generalization of the modelling approaches of monitoring.

In summary:

- We plan to explore further the modelling of the Monitoring building block, by analyzing the cross influences and tradeoffs of security and safety. We are particularly interested in analyzing the models that allow us to study the sensitivity or specificity of the monitoring building block, with regard to either safety or security incidents, and the interplay between the two types of incidents. This would allow better decisions to be made on the configuration of the monitoring systems by providing the decision makers with an analysis method for quantitatively assessing the trade-offs between security and safety requirements.
- We plan to explore the possibility of obtaining some real data for one of the use cases in SESAMO, preferably the Avionics use case which we have been analyzing so far. This will allow us to validate modelling assumptions; primarily the ones summarized in the previous bullet point, but also some others made as part of the work in WP2.
- We plan to do a more thorough analysis of the avionics use case with a more detailed catalog of examples of synergies and trade-offs of security and safety requirements that may result from the use of monitoring in Avionics.
- We aim to provide a more detailed description of a prevention monitor model able to fulfil safety and security requirements. We plan to assess the applicability of this model when used for the e-health use case in which the monitor has to guarantee confidentiality, integrity and availability of sensitive data.

4.3 PARTITIONING

4.3.1 Analysis in Avionics Use-Case

4.3.2 Analysis in use-case avionics

4.3.2.1 Use case related description of problem

SESAMO is dedicated to bringing security to dependable real time systems that are governed by domain specific process and architecture standards.

The avionics system:

Figure 4-12 shows an Integrated Modular Avionics (IMA) system view from an Airbus 380\textsuperscript{12} with a focus on the partitioned system network AFDX (ARINC664). It shows different system parts like Engines, Cabin, etc. and how they are connected by a double redundant

\textsuperscript{12} Albert Benveniste (INRIA-IRISA, Rennes / COMBEST)
(the blue and the red) AFDX network system. The big arrow through IMA system shows a possible information flow between components through the AFDX network. Figure 4-13 shows a part of the tool support for an AFDX system from a SESAMO partner.

Figure 4-12: IMA / AFDX system view and communication

Figure 4-13: Integrated Modular Avionics Configuration Tool IMACT ©SYSGO

IMACT is able to generate the configuration for all components of the modelled IMA system, the certification artefacts according DO178C/DO254 standards, and apply different analysis (e.g. consistency, reliability etc.).
Partitioning and Safety

Partitioning from a safety perspective focuses on space and time partitioning.

Space partitioning targets at:

- Protection of program, data, registers and dedicated I/O
  - Examples like: Persistent storage / memory is writable by one partition
- Temporary storage and registers of a partition are saved and not contaminated, when control is transferred

Time partitioning targets at:

- Protection of processing and communications bandwidth assigned to a partition,
- A partition’s access to a prescribed set of hardware resources for a prescribed period of time is guaranteed,
- The order of execution between communicating partitions is consistent in each execution frame, resulting in a defined execution of the threads contained in the correlated process of the partition,
- Every thread gets allocated all its requested CPU resources every period.

Partitioning and Security

For the use-case we consider the off-board and on-board aircraft (security) network domains, see Figure 4-14. Here connectivity “off-board” comprises passenger connectivity, automatic content refreshes (maps, entertainment systems), and maintenance activities in addition to traditional control tasks (e.g. for air traffic management). Security further concerns on-board systems and their separation and information flow. The standard ARINC811 describes aircraft operations and maintenance considering security aspects. Figure 4-14 depicts aircraft network security domains, major aircraft system, and access properties (closed, private, and public) and users of domains. The standards deal with these mixed levels of security systems.
Figure 4-14: On-board aircraft network domains (© ARINC-811)

Figure 4-14 shows the security-related environment of the specific use case. In parallel to IMA (ARINC 653), Multiple Independent Levels of Security/Safety (MILS) are proposed. Like IMA, MILS is also based on partitioning with a focus on security related attributes. MILS focuses on a system architecture built out of components where components are analyzable from security construction perspective. In a nutshell, MILS is a high-assurance security architecture based on the concepts of:

- Separation and controlled information flow,
- Implemented by separation mechanisms that support both untrusted and trustworthy components,
- Ensuring that the total security solution is non-bypassable,
- Ensuring that the total security solution is evaluable, and
- Always invoked and tamperproof.

The EADS use case looks at a MILS system from two perspectives. The first perspective addresses the controlled information flow. Controlled information flow means that data only flows in a controlled way (using a traffic flow monitor, also called traffic filtering device, gateway, …). From a second perspective, the gateway itself is based on partitioning: partitioning, which is guaranteed at hardware level (memory management units) and by the operating system (deterministic schedule and dispatch management, task and memory management).
Narrowing the use-case – secure gateway for avionics\textsuperscript{13}

Figure 4-15 depicts an integrated system with two security domains communicating unidirectional. For the bilateral communication, the gateway components in one direction need to be “replicated” for the other direction.

![Figure 4-15: Integrated gateway software architecture based on IMA and MILS](image)

Each security domain can comprise one or more partitions. The integrated secure gateway, in the center of the picture comprises two partitions belonging to different domains. The picture indicates that the partitioning properties of the operating system build the foundation of the partitioning arguments of the gateway and its communication checking properties.

4.3.2.2 Analysis of safety and security cross influences

Partitions bring in on the safety side overhead some overhead like delays when context-switching. It is possible proof a guaranteed maximum latency thresholds taking into account all intra- and inter-partitions communications possibilities of a schedule policy. From a safety perspective, a certain behavior of a partition can be assumed dependent on its safety level. Also assumptions can be made about failure behavior for safety (e.g. if some code coverage is used then the code implementing the partitioning can rely on this code coverage). On the other side, for security no assumptions about the code behavior can be made (e.g. it must be assumed that the program running issues an invalid command/operation and the OS needs to deal with the “cleanup” within its time slot OR it at least needs to be taken into account in the analysis.).

The safety and the security approaches rely on the underlying partitioning of the operating system or the network. There is interaction between safety and security in this topic because if partitioning is attacked from a security perspective, it has also influence on the safety.

Security mechanisms include checks at start-up and checks during operation. The latter are the more interesting ones and they have to run in parallel to functional threads inside a partition. In doing so, security mechanisms must not destroy any of the partitions principles. On the contrary: security has to be part of the overall IMA System which is made up of parti-
Partitions and is subject to scheduling policies and timing principles. So security software has to be checked in real-time (i.e. has to follow real-time requirements) and has to fit in strong cyclic scheduling of partitions. All security related software within a partition has to be certified to the criticality level of that partition at least. An example is: A watch dog detecting intrusions in a level A partition has to be certified on level A or it has to run outside that partition.

So to recapitulate:

- Partitions for safety means context switches; partitions for security means additional context switches,
- There can be attacks on stored context information,
- Adding security to safety costs resources (of any kind: CPU, BUS, I/O, …), but that is not a problem in principle,
- The partitions in principle provide a scheme for safe real time systems, timing principles are not to be violated by any added security principles and code,
- Security software may require real time checks and should fit into strong cyclic scheduling of partitions to alleviate partitioning
- All software running in a safety level A (or B, C, …) partition has to be certified at least to level A (or B, C, …, respectively),
- The safe software and the security software must not run in the same IMA partition, so they can be certified to different safety levels, but:
- When safe software and the security software are running on different partitions in the same security domain, they only can use the inter-partitions communication of an IMA system as described in the avionics use case (which must not be a problem), for communication between different security domains a gateway is needed additionally.

**4.3.2.3 Preliminary results and next steps**

We have analyzed the avionics use-case (D1.1) from the point of view of usage of the building block “Partitioning” in the context of a possible modeling of this building block in IMACT tool. In this preliminary analysis we have identified that the description and approached used in these three parts are compatible as wells that the current state of the building block “Partitioning” and its modeling is sufficient for the avionics use-case.

Our next steps are more detailed analysis of the synergies of safety and security with respect modeling and standardization requirements as well as working out on synergies with the railway use-case.

**4.3.3 Analysis in railway**

**4.3.3.1 Use case related description of problem**

The ESSI (Embedded Safety and Security Interface) is a device which provides safety and security communication among safety related applications through an open communication system. Safety related applications are placed in a closed communication system. The definition of open/closed
communication system is involved in standard EN 50159. From ESSI functional aspect it is very important that in closed communication systems the risk of unauthorized access is inappreciable.

Main

1) Create a secure communication channel through an open communication interface. The main aim of the secure channel is the protection of application data flows against the loss of data integrity and authenticity (in terms of EN 50159 – message corruptions and message masquerade) using cryptographic methods. The protection against loss of data confidentiality is optional. The possible protection methods are using hash functions (ISO/IEC 10118-1, 2), message authentication codes (ISO/IEC 9797-1, 2) or digital signatures (ISO/IEC 9796-2, 3).

2) Separate the closed and the open transmission system. Attacks against the ESSI from the open transmission system must not propagate to the applications in the closed transmission system. The worst case scenario is loss of communication between the ESSI partners.

The ESSI units always work in pair. The pair of appropriate ESSI units establishes a session that will transmit data between safety related applications. This session is based on UDP/IP. The ESSI is transparent for safety related systems for which ESSI provides the security channel (In other words, the ESSI adds a new transparent security layer to the communication stack between two safety related applications). The ESSI assembles received bytes/packets from Local Communication Interfaces (Figure 4-167) into packet and add to packet protective code e.g. MAC (Message Authentication Code) and sends them through an Open Communication Interface (Figure 4-16) to the partner ESSI. The partner ESSI receives the packet on the Open Communication Interface and checks the protective code of the packet (authenticity and integrity validation of received packet), if the check is successful the packet is disassembled and the parts are distributed to the appropriate Local Communication Interfaces.

![Generic use case of ESSI](image-url)
ESSI can be decomposed into following high level components (Figure 4-16).

3) Local Communication Interfaces The main function of this block is providing communication interface for safety related applications. This functional block is not unlike the others generic, but their particular implementation depends on safety related application for which provide communication interface (kind of physical and link layer). But it is possible to define main functions which are common for all kinds of Local Communication Interfaces. These functions are:
   - Receiving data from safety related application.
   - Sending data to Multiplexer/De-multiplexer.
   - Reading data from Multiplexer/De-multiplexer.
   - Sending data to safety related application.
Considered physical interfaces are:
   - Ethernet.
   - CAN.
   - RS-232.

5) Multiplexer/De-multiplexer.
   This Block provides following functions:
   - Reading data from Local Communication Interfaces.
   - Assembling data received from Local Communication Interfaces.
   - Adding protective code item into assembled packet.
   - Sending packets to Open Communication Interface.
D2.1 Specification of Safety and Security Mechanisms

- Reading packets from Open Communication Interface.
- Checking protective code.
- Disassembling packets into parts corresponding to individual safety related applications.
- Sending disassembled parts to Local Communication Interfaces.
- Providing a finite state machine for establishing/cancellation of the secure relation with partner ESSI.
- Optionally provide encryption/decryption of packets.

6) Open Communication Interface.
Block provides following functions:
- Reading packets from Multiplexer/De-multiplexer.
- Sending packets to partner ESSI.
- Receiving packets from partner ESSI.
- Sending packets to Multiplexer/De-multiplexer.

7) Service interface/Monitor.
Block provides following functions:
- Providing information about actual state of ESSI for maintenance.
- Provide access to logs for maintenance.

Figure 4-18: Component diagram of ESSI (high level view)
NOTE: Figure 4-18 the application data flows are drawn as bidirectional, but in fact data flow between two functional blocks involves two independent unidirectional data flows with opposite direction.

### 4.3.3.2 Use case requirements to Partitioning

The main aim of partitioning is to segregate mixed-criticality components of a system into different domains. For each domain a set of system resources is assigned. The main types of resources are following:

1. Space e.g. memory, I/O devices, CPU, Buses.
2. Time e.g. CPU time, bus bandwidth.

As shown in Figure 4-18, in ESSI both space and time partitioning are used. Each high level component of ESSI has its own space and time domain.

From aspect of effect on safety of ESSI is important assignment of following resources:

- Memory. Assignment of memory is a protection against unintentional overwrite data by another component.
- IO devices. Each component has its own associated I/O device (I/O devices are not shared between components in ESSI use case). Assignment of I/O device is a protection against unintentional using of device by another component.
- Time. Protection against overall overloading ESSI by overloading of one component.

Another effect on safety is fault isolation. Each fault appearing in one component should not be propagated to the other component. E.g. if a fault occurs in the component Local Communication Interface 1, communication is interrupted only for safety related applications 1, other communications should be still working.

From security aspect ESSI use case has the requirement, that the security attack to one component should not be propagated to other components. We suppose that security attacks only come from the Open Transmission System. This attack should not be successfully propagated to the Multiplexer/De-multiplexer component, because this component provides services for protection of application data flow against loss of data integrity and authenticity and loss of trust of this component could lead to loss of trust of secure communication channel between two ESSI. If a security attack is propagated to the Local Communication Interface, it leads to a threat for the Closed Transmission System.

### 4.3.3.3 Analysis of safety and security cross influences

Because the component Multiplexer/De-multiplexer also provides function related to security, it is dedicated for this component’s individual domain (partition). Security related functions of this component are:

1. Creating and checking protective code which is added to packets sent through the Open Transmission System.
2) Providing a finite state machine for establishing/cancellation of the secure relation with the partner ESSI.

In other words, this component has access to a shared secret which is necessary for creating and checking protect code and for establishing a secure relation between ESSI partners. This component hasn’t direct access to any physical communication interfaces (it is encapsulated in system).

This improving of security could lead to the cross influence to safety by adding additional delay to the communication between the safety related applications for which ESSI provides a security channel.

This influence was identified in Section 2.16 as “Hard to prove to keep guaranteed maximum latency threshold taking into account all intra- and inter-partition communication possibilities of a schedule policy”.

4.3.3.4 Preliminary results and next steps

In the ESSI use case partitioning is implemented via PikeOS on P3041/P2041 QorIQ HW platform Figure 4-19.

The PikeOS is a micro-kernel based operation system targeting to safety critical embedded systems which provides partitioning and virtualisation. The partition management of PikeOS controls the assignment of system resources and protects all partitions against each other. From railway aspect it is important that Pike OS is certified according to following standards IEC 61508, EN 50128, EN 50129.

In the preliminary analysis of the building “Partitioning” for railway use-case we have identified the following issues:

1) Description of inter-domain (inter-partition) communication. Consideration about its impacts on safety and security. Consideration about atomicity of services provided by inter-domain communication (e.g. writing/reading to/from shared memory etc.).

2) Consideration about propagating of HW events into partitions. E.g. what are the possible consequences of the partition, when too many interruptions are generated from device which isn’t assigned to this partition?
4.3.3.5 Summary
In this contribution we present using of building block “Partitioning” in ESSI use case. We present how to this building block satisfies requirements derived from the concept of ESSI and related standards (ARINC 653, EN 50129). The function block “Partitioning” described in Section 2.16 provides for the ESSI use case one of the base concepts for definition of high level components.

4.4 REDUNDANCY AND DIVERSITY
The essential challenge in assessing the combination of safety and security is in the greater variety of threats or faults to be considered, complicating the combinatorial analysis of redundancy (“is there an appropriate element of redundancy against each failure condition for which one is required?”) and consequently that of probability of common failure. With respect to the former, the model of a redundant/diverse component may need to be extended with the additional interfaces through which adversaries may operate; with respect to the latter, with separate parameters for the probabilities of common failure with respect to different causes – accidental or malicious – with possibly substantially different values. The hypothetical example that follows illustrates the problem in general; the following subsections outline the studies of redundancy and diversity envisaged in the SESAMO use cases.

4.4.1 Example (1-out-of-2 redundancy/diversity):
Let us consider a duplicated controller (that is, made up of two, possibly diverse, replicas or channels) with an operation/maintenance interface accessible via a network.

Let us also assume the architecture to be such that for an accident to happen it is necessary that both channels exhibit failures (erroneous behaviours) that would cause an accident if that channel were the whole controller (we will call these critical channel failures). That is, the system is safe as long as at least one channel’s behaviour satisfies a safety condition.

Figure 4-20 Duplicated control system with different ...
4.4.1.1 System-level safety requirements and role of security

If one out of two channels fails then the remaining channel is able to detect the failure and trigger a transition to a failsafe state.

The channel failures can be caused by:

- Accidental hardware failures;
- Accidental channel failures caused by design faults in the channel software, activated by normal operational/maintenance signals/actions via normal interface(s)
- Channel failures due to software faults introduced by an adversary
  - During the design/implementation
  - During maintenance (rootkits) due to physical access to the channel
  - Via the network configuration/maintenance interface
- Hardware design faults introduced by an adversary via physical access to the unit (rewiring)
- Actions by an adversary (via normal operational/maintenance interface(s)) exploiting accidental design hw/sw faults
- Combinations of 1 – 5

From the viewpoint of a safety requirement “no accident shall be caused by this controller”, the chances of it being satisfied during operation depend on the probabilities of failures of both channels, due to any combination of these causes.

Thus the security viewpoint arises in that an adversary may produce an accident by means 3-6 in the above list. This determines integrity requirements for the channels: we want the adversary not to be able to cause these failures. Violations of integrity matter in that they may cause an accident: the adversary may succeed in making both channels behave in unsafe ways, or it may at least make one of the channels unable to react properly when the other one fails due to accidental causes: it would make the system effectively non-redundant.

So, introducing security consideration does not change the characterisation of the system as a 1-out-of-N system: the system only fails if all channels fail, for any reason. However, the probabilities of common critical failure may be different for the three possible combinations: both channels fail for accidental reasons, both due to malicious action, or one for accidental reasons and one due to malicious action. For instance,

- there may be very effective precautions against common failure due to accidental failures, but some common vulnerability affecting both channels, so that the probability of an adversary causing critical failures of both channels is not much lower than that of it causing critical failure of one
  - the containment barriers between the channels may be effective for preventing propagation of accidental failure but be permeable to attack (e.g., an attacker might be able to cause a compromised channel to issue outputs on a cross-comparison inter-
face that cause failure of the other channel, or disrupt a communication protocol, and would be an incredible failure mode if only accidental causes are considered)

- the channels might also have confidentiality vulnerabilities allowing an adversary to learn from one channel some information that allows an attack to cause (or increase likelihood of) failure of the other channel.

### 4.4.1.2 System-level security requirements

So far we have considered “security for safety”, i.e., the need to prevent adversaries from endangering the reliability of safety functions. If the system also has security-only requirements, different considerations may apply. We choose as an example confidentiality requirements.

Confidentiality requirements mean that both channels contain some asset of interest (A) and though with no direct effect on data/system integrity and thus safety, the adversary access to A would be a loss. As an example, we consider the case that the asset is common to both channel. Confidentiality requirements may be:

- completely independent of safety of the individual system compromised, e.g. being aimed at safeguarding intellectual property in the controller’s software in order to prevent indirect losses unrelated to safety of the system under consideration;

- but they may also be due to indirect effects on safety, which depend on the details of the architecture and the controlled system: e.g., access to input/output data or to the code might allow the adversary to devise better attacks on the controller; or on the controlled system by exploiting weaknesses of the control algorithm; or an attack by spoofing the control outputs of the controller. All these scenarios are still related to safety, but within an extended system boundary

- likewise, they may arise from a concern that by learning about details of a controller in a certain application context (controlled system), the adversary may gain information for attacking similar controllers in a different contexts.

With respect to the confidentiality requirement, if for every channel there is a likelihood of the adversary getting access to it, and the adversary will spend the same amount of effort on each channel regardless of their number\(^\text{14}\), then the system of interest behaves as “series” or “2 out of 2” system: compromising one channel compromises the whole. Adding more channels, while decreasing risk

---

\(^{14}\) Note that this is assumption about the strategy of the adversary depends on their cost function. This strategy describes for instance an adversary that has ample resources but needs to contain the bandwidth of probes on each channel to avoid detection; or one who will simply replay a standard set of attacks on any channel available. Notice that even if the adversary had to divide a fixed amount of effort over multiple channel, this scenario would still lead to redundancy damaging confidentiality, although assessing by how much would be more complex.
due to accidents, would at the same time increase indirect risk due to confidentiality violation. A trade-off arises between confidentiality and safety; there is a number of channels that minimises total risk from direct or indirect losses. The optimum degree of redundancy depends on the combination of the adversary’s strategy and the particular loss functions associated to the various loss events. (A similar trade-off may arise with respect to the channel-level, directly safety-dictated confidentiality issues mentioned at the end of the previous section.)

4.4.1.3 Summary
This example system and a set of example scenarios related to it, among the many that are possible even for such a simple architecture as outlined, highlight a few essential points for the study of synergies (e.g. “security for safety”) and trade-offs between security and safety considerations. These will be studied on concrete examples in the SESAMO use cases described below.

4.4.2 Avionics use case – Secure Gateway
The EADS use case centres on a Secure Gateway based on MILS principles in an IMA context. This does not directly use redundancy and diverse replication measures itself in its core, but rather focuses on the building blocks “partitioning”, “traffic filtering”, etc.

However, possible extensions leveraging the redundancy building block principles are thinkable and are briefly discussed in this section to stimulate discussions in the SESAMO project and experimentally using the components in a pure research fashion for achieving additional scientific insights. As discussed above, redundancy without the use of at least some level of diversity seems to be of limited value to increase security properties of a system as any attack get leverage weaknesses (vulnerability) in all replicas. Hence, only redundancy with diversity is discussed below.

**System diversity.** Given the strength of any approach diversity approach relies on the degree of independence of the two or more diverse replicas, diversity at system level by deploying two completely different versions of a secure gateway design seems to be most promising. In such a case, the one of the approaches towards solving secure gateways may even not be architected leveraging MILS principles.

System level diversity comes of course with the associated costs. System-level diversity is not uncommon in the safety arena. Most promising and well known in the avionics domain are the flight control architectures of large transport aircraft like Boeing’s B777 and Airbus’ A320/A330/A340/A380.

In case system diversity is thought after the different gateway replicas would be deployed in series to guarantee the integrity of the critical domain. Any parallel architectures (i.e. two security domains are connected by two or more diverse gateways) are likely the wrong approach as they allow a bigger protection surface and aerospace does not need availability is urgently as integrity.

**Diversity of gateway components.** Another way to introduce diversity in the secure gateway use case would be on the component level. Filters could be implemented independently and in a diverse way. These would need to be arranged logically in series to guarantee integrity of the higher security domain. Trade-offs in the degree of redundancy and adjudication principles could be studied, in view of the dual requirements of guaranteeing integrity and containing the frequency of “false alarms”.
This mostly would address the development of the filters and less the architectural principles of MILS and its partitioning properties. In case filter components are use, they need to be arranged in series.

**MILS architecture component diversity.** Similarly to gateway components diversity, any replicated gateway could equally use different MILS components, like hardware (e.g. computing platform) or software (e.g. OS). This ensures correct operation of traffic filtering at system level if multiple gateways are used (in series). This mostly would address assurance of partitioning properties.

**Process diversity.** In addition to replication, just leveraging different development processes and teams can also lead to an improvement of security and safety properties.

### 4.4.3 Analysis in Railway Use-case

#### 4.4.3.1 Use case related description of problem

ESSI (Embedded Safety and Security Interface) is a device which provides safe and secure communication among safety related applications through an open communication system. Safety related applications are placed in closed communication systems. The definition of open/closed communication system is in standard EN 50159. From the viewpoint of functional security of ESSI, it is very important that in the closed communication system the risk of unauthorized access is negligible.

According to the above description, ESSI has the following main function (Figure 1):

4) Creating a secure communication channel through an open communication interface. The main aim of secure channels is provide protection of application data flow against loss of data integrity and authenticity (in terms EN 50159 – message corruptions and message masquerade), using cryptographic methods. Protection against loss of data confidentiality is optional. The possible protection methods use of hash block codes (IEC 10118-1,2), MAC (IEC 9797-1,2) or digital signature (IEC 9796-2,3).

The Building Block Redundancy and Diversity is considered for improving the following properties of secure channel:

- Improving availability of ESSI function (data transfer among safety related applications). This is based on creation of two secure channels between two ESSI, which simultaneously transfer the same data (Their contents are not compared.); the first correctly delivered packet is used.
- Improving of safety properties of data transfer between two ESSI, more precisely improving detection of data integrity loss. This is based on creation of two secure channels between ESSI, which simultaneously transfer the same data, and their contents are compared by receiver ESSI.

The choice between improving these properties of secure channel depends on used voting mechanism (1oo2 – availability, 2oo2 – data integrity loss detection).

5) Separating the closed communication systems from open communication systems. The security attacks on ESSI, coming from the Open Transmission System, must not be propagated to applications in the Closed Transmission System. The worst case scenario is loss of communication between ESSI partners.
For simplification of the description of the ESSI function, we choose between two voting mechanisms 2oo2 (data integrity loss detection) or 1oo2 (availability).

The ESSI units always work in pair. The appropriate pair of ESSI units establishes a session that will transmit data between safety related applications. This session is based on UDP/IP. The ESSI is transparent for safety related systems, for which ESSI provides secure channels (In other words, ESSI add a new transparent security layer to the communication stack between two safety related applications). The ESSI assembles received bytes/packets from the Local Communication Interfaces into packets and creates replica of assembled packets (Figure 2). ESSI adds to packets (original and replica) protective code, e.g. MAC (Message Authentication Code), and sends them through the Open Communication Interface to the partner ESSI. The Open Communication Interface provides separate channels for original and replica packets (Figure 2). The partner ESSI receives the pair of packets (original and replica) on the Open Communication Interface (from both the channels dedicated for original and replica), checks the protective codes of the packets (authenticity and integrity validation of received packet) and if the goal is improving ability to detect loss of data integrity checks, then if the content of the two packets is the same, or if the goal is improving availability only first apparently correct packet delivered, out of the “original and replica” pair, is used. If the checking is successful, the original packet is disassembled and the disassembled parts are distributed to the appropriate Local Communication Interfaces. Negative result of checks should instead lead to disconnection of the secure channels (both the channel for original packets and channel for replica packets).
ESSI can be decomposed into the following high level components (Figure 2):

8) Local Communication Interfaces.
9) Multiplexer/De-multiplexer.
10) Open Communication Interface.
11) Replicator.
12) Service interface/Monitor.

4.4.3.2 Analysis of safety and security cross influences

The BB redundancy and diversity could be used for following goals:

1) For improving availability of ESS function (secure data transfer) by using voting mechanism 1oo2.
2) For improving safety properties of secure channel between two ESSI (ability to detect loss of data integrity during transmission in secure channel).

In the case that both channel (for original and replica packet) use the same mechanism for data transfer, authenticity and integrity protection only uses Redundancy (this does not depend on the selected voting mechanism).

The strength of redundancy could be improved by using diversity. Possible target of diversity in secure data transfer are:
1) Using different protective codes in the two channels.
2) Using different shared secrets in the two channels.
3) The replica assembled packet could be the bit-complement of the original.

Possible cross influences are following:
1) This improving of safety could lead to cross influence on security by extension of the attack space (two channels instead of one channel).
2) This improving of safety could lead to cross influence with another safety property, real-time behavior of communication between two ESSI (adding delay). Because we are not able to ensure the same delivery time for delivering the original and replica packets, we need to define the time interval (time window), in which ESSI will wait for delivering of packets with bigger delay.

4.4.3.3 Preliminary results and next steps
The goal of improving the data integrity loss detection could be probably achieved by another method. The main goal of Diversity is identification of the loss of data integrity. For instance, this goal could be achieved by a method using protective code based on cryptographic method, which are described in building block Authentication. The question arises:

1) What are the trade-offs and synergies between BB Redundancy, Diversity and BB Authentication (or Integrity protection).
2) What is the appropriate pair of different protective codes which could be used diversified secure channels.

The goal of improving availability could be achieved by Redundancy using 1oo2 voting and this could be a major contribution of this BB for the railway use case.

We consider using Redundancy and Diversity as optional methods. An interesting idea is the possibility to change voting model (of course in system definition, not dynamically) e.g. model 2oo2 improving ability to detect loss of integrity in secure channels and 1oo2 improving functional availability.

4.4.4 Analysis in Industrial Drives use case

4.4.4.1 Use case related description of problem
The Industrial Drives use case is a typical motion control application out of the field of industrial automation and control. Motion Control products cover a large variety of variable frequency inverters for synchronous and asynchronous motors ranging from standard electric motor systems and servomotors for Motion Control applications including linear and torque motors to motors for use in hazardous explosion areas, to high voltage, DC and customized electric motor systems.

This use case focuses on a generic commercial motion control platform solution for permanent magnetic synchronous motors (PMSM). Typical application is within e.g. tooling machines. Since this application is based on a generic control reference platform it is also possible to target application in similar domains like Electric Drive Trains in automotive.
Within most safety relevant systems replication of critical components is a common pattern. Within the current use case this mechanism is highlighted in two places, on the one hand SW oriented replication in the context of sensor data transmission, on the other hand pure HW replication of the PWM modules, see Figure 4-23.

Figure 4-23: Replication within the Industrial Drives Use Case

1. Replication of the authenticated Encoder Messages:
Within the given application the Replication Building Block is applied for hardening the sensor data transmission path between the rotary sensor mounted on the motor axis, and the processing SW algorithm running on the embedded processor within the FPGA.

Above Figure 4-23 illustrates the simplified approach where the sensors and the associated encoder interfaces are tripled. The related SW component reads three values instead of one and autonomously takes a majority voting decision on the received data values.
This admittedly on the first glance relatively unchallenging example was selected because interesting relations to other SESAMO building blocks are given. The modelling of those interdependencies between different safety and security building blocks will a central achievement of the SESAMO methodology.

- **BB Authentication and Real-time Communication:**
  The Encoder Sensor messages will be authenticated by MAC, which is already analyzed in Section 2.18. The resulting delay caused by this will again be increased by the voting algorithms.

- **BB Plausibility Checks:**
  The received values from the 3 different sensors will slightly be slightly different which makes it necessary to have a tolerant voting combined with plausibility checks.

2. **HW Replication of the PWM Modules:**
In contrast to the previous the PWM modules are replicated purely within HW. Here three identical instances are present which process identical input stimuli. Identical output is expected and direct comparison reduces the voting. Main benefit of this measure regarding safety is that technology related failures, caused by radiation or aging can be detected, and handled. Main intention for analyzing this aspect within SESAMO is that there are commercial FPGA synthesis tools available, that can automate the replication of individual blocks within a RTL design. The challenge within SESAMO is to analyze if it is possible to integrate this into the SESAMO flow respectively the tool chain.

4.4.4.2 **Analysis of safety and security cross influences**
1. **Replication of the authenticated Encoder Messages:**
   Also for this mainly SW-oriented replication approach common safety security cross-influence pattern are suitable:
   
   - The additional computation effort cause additional delay that can badly influence the real-time behaviour of the system.
   - Also true is that the replication widens the attack space, and generally increases the vulnerability of the system.

   Much more interesting is the following aspect that arises when the authentication block is replicated in a diverse manner. In concrete this is the case when the three sensor instances authenticate their messages with different keys, or in the extreme, with different algorithms.

   Assuming that an attack is aimed to introduce fake sensor data this replication proportionally increases the attacker’s efforts, and consequently raises system security in the same ratio. So an attacker has to figure out several different keys and also the different underlying algorithms. It is clear that diverse replication also affects the complexity of the related SW parts and needs additional computation time....
Parameters:

- Number of instances
- Decision Principle (2003, 1002, ….?)
- Added delay for voting SW

Safety Analysis Method:

- Calculation of PFD (probability of failure on demand)

Security Analysis Method:

- Classification of the Authentication Algorithms

2. **HW Replication of the PWM Modules:**
   There is no significant security impact besides the general cross-influence pattern that the attack area is wider. For this reason no further cross-influence analysis is conducted in this place.

### 4.4.4.3 Preliminary results and next steps

1. **Replication of the authenticated Encoder Messages:**
   **preliminary results:**
   - Within the related Artemis ACROSS project voting services were implemented, which may be applicable also for SESAMO.

   **next steps:**
   - Combine the related building blocks, analyze and extract relevant properties from the implementation view for a more abstract modelling view.

2. **HW Replication of the PWM Modules:**
   **preliminary results:**
   - The RTL design of the FPGA system is already available, especially including the VHDL source code of the PWM module.

   **next steps:**
   - A research on available RTL replication tools will be conducted, focusing on the question how the tools can be integrated into the SESAMO flow.
4.4.5 Preliminary Conclusions, Outlook and Generalisation Potential

The models and methods that have led to the preliminary conclusions in this chapter and are to be applied to the detailed analyses to follow can be classified roughly as:

- descriptive models of how a redundant and/or diverse component/system is built, to be more or less detailed depending on the intended analyses

- models of deterministic dependency relationship, e.g., “this service will be delivered correctly provided only this subset of possible component failures occur” and “this cause has the potential to cause failures of these multiple components”. These models are involved in deterministic analyses as e.g. in criticality analysis and common may be built explicitly or remain implicit when performing analyses, e.g. “safety criticality analysis” and “dependent failure analyses” as defined by standards

- probabilistic models, meant mostly to complement the deterministic ones with statements like “given that the failures of these two components would cause a system failure (as documented in the deterministic models), what can we say about the probability that this scenario actually occurs?” and thus allow reasoning about design trade-offs. These models are typically combinatorial models - this chapter has referred to results developed in previous research that are suitable for extension here, as well as a set of worst-case results developed in SESAMO

In this document, the cross-influences between safety and security have been discussed at the level of the first two categories of models, with indications of how they will be reflected into the probabilistic models. The next steps will be detailed analyses of the safety and security aspects and their interdependencies.

We expect that the set of analyses envisaged will prove sufficiently varied to satisfy the need for generality of solutions in SESAMO and outside it. Indeed, the proposed applications in SESAMO use cases cover both situations in which the effect of redundancy and diversity - introduced for safety reasons - may create security risks that indirectly affect safety; the case (4.4.4) in which redundancy and diversity are introduced for the sake of security; and a case (4.4.3) in which the same system may be configured with different tradeoffs between protecting integrity and reducing delay. This should be a rich enough set of examples to ensure that the analyses developed can cover a broad set of needs in the various architectural combinations of redundancy and diversity, with alternative types of “adjudication”, targeted at various goals of security and safety at the overall system level. From the viewpoint of the complexity of the analysis, these examples are reasonably simple and yet present the main requirement to analyse the attack interfaces, to deal with the combination of accidental and maliciously caused failures in the component replicas, and to consider the different sources of probabilistic information that need to be fed into analysis so as to document the trade-offs for the designer.
5 CONCLUSIONS

This deliverable provides a partial view on the SESAMO tool chains. We have presented first results for the three major steps for SESAMO tool-chains development life-cycle: what (to model), how (to model), and usage. All these steps are made from the point of view of “building blocks”, which are the basic elements and are the “what’s” of the tool-chains. The mentioned three steps are:

1. Identification and definition of the basic building blocks of the tool-chain

   The building blocks are the main elements of the SESAMO tool chains. In this document we defined their major functionality, interfaces, influences of safety and security as well as identified trade-offs for safety and security for the BB functionality. We made this analysis in a semi-formal way to guarantee modular and flexible architecture of the tool-chains.

2. Presentation of the modelling approaches for selected building blocks

   These results describe how in practice the modelling could look like. The purpose is to give the consortium different view on modelling approaches to identify synergies, which are necessary to build joint tool chains. These approaches span across different standards and domain usage to highlight the difference, and thus, facilitate search for synergies.

3. Usage of building block within a use-case

   The final step in this deliverable is identification and preliminary usage and preliminary analysis of the building blocks in the SESAMO use-cases. We showed how building blocks reflect the functionality and requirements needed in use-cases and collected feedback on the BB description and modelling. This feedback will be used in D2.2 to finalise the analysis and prepare input for integration and implementation phase of the project.
6 REFERENCES

[10] ISO/IEC 18033 (all parts), Information technology - Security techniques - Encryption algorithms
Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function


http://www.freepatentsonline.com/DE102008026729.html


[60] www.csr.city.ac.uk/diversity
D2.1 Specification of Safety and Security Mechanisms


## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1oo2</td>
<td>voting mechanism 1 out of 2</td>
</tr>
<tr>
<td>2oo2</td>
<td>voting mechanism 2 out of 2</td>
</tr>
<tr>
<td>AAL</td>
<td>Ambient Assisted Living</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>ARINC</td>
<td>Aeronautical Radio, Incorporated</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>ASIL</td>
<td>Automotive Integrity Level</td>
</tr>
<tr>
<td>ATV</td>
<td>Automated Transfer Vehicle</td>
</tr>
<tr>
<td>AUTOSAR</td>
<td>AUTomotive Open System ARchitecture</td>
</tr>
<tr>
<td>BB</td>
<td>Building Bock</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CBC-MAC</td>
<td>Cipher Block Chaining Message Authentication Code</td>
</tr>
<tr>
<td>CBC</td>
<td>Cipher Block Chaining Mode of Operation</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSMA/CR</td>
<td>Carrier-Sense, Multiple Access with Collision Resolution</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier-Sense Multiple Access</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DER</td>
<td>Distributed Energy Resource</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DSA</td>
<td>Digital Signature Algorithm</td>
</tr>
<tr>
<td>DSO</td>
<td>Distributed System Operator</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>ECC</td>
<td>Elliptic Curve Cryptography</td>
</tr>
<tr>
<td>ECC</td>
<td>Error Correcting Code</td>
</tr>
<tr>
<td>ECC</td>
<td>Error Correction Code</td>
</tr>
<tr>
<td>EMAC</td>
<td>Encrypted MAC</td>
</tr>
<tr>
<td>ESSI</td>
<td>Embedded Safety and Security Interface</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-programmable gate array</td>
</tr>
<tr>
<td>GOOSE</td>
<td>Generic Object Oriented Substation Events</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hash-based Message Authentication Code</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>IED</td>
<td>Intelligent Electronic Device</td>
</tr>
<tr>
<td>IMA</td>
<td>Integrated Modular Avionics</td>
</tr>
<tr>
<td>ISO/IEC</td>
<td>International Organization for Standardization/International Electrotechnical Commission</td>
</tr>
<tr>
<td>ISO/OSI</td>
<td>International Organization for Standardization/Open Systems Interconnection</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standardization Organization</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>MILS</td>
<td>Multiple Independent Levels of Security</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability of failure per demand</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PMSM</td>
<td>Permanent magnetic synchronous motors</td>
</tr>
<tr>
<td>Acronym</td>
<td>Full Form</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>PUF</td>
<td>Physically Uncloneable Function</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse width modulation</td>
</tr>
<tr>
<td>RFC</td>
<td>Request For Comments</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>RSA</td>
<td>Rivest, Shamir und Adleman Signature Algorithm</td>
</tr>
<tr>
<td>RTCP XR</td>
<td>RTP Control Protocol Extended Reports</td>
</tr>
<tr>
<td>RTCP</td>
<td>RTP Control Protocol</td>
</tr>
<tr>
<td>RTL</td>
<td>Register transfer level</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>RTU</td>
<td>Remote Terminal Unit</td>
</tr>
<tr>
<td>RTOS</td>
<td>real time operating system</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control and Data Acquisition</td>
</tr>
<tr>
<td>SEooC</td>
<td>Safety Element out of Context</td>
</tr>
<tr>
<td>SW</td>
<td>software</td>
</tr>
<tr>
<td>SysML</td>
<td>Systems Modeling Language</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TDEA</td>
<td>Triple Data Encryption Algorithm</td>
</tr>
<tr>
<td>UDP/IP</td>
<td>User Datagram Protocol/Internet Protocol</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>WCET</td>
<td>worst case execution time</td>
</tr>
<tr>
<td>XACML</td>
<td>Access Control Markup Language</td>
</tr>
</tbody>
</table>