PREparing SEcuRe VEHICLE-to-X Communication Systems

Deliverable 2.4

System Integration Report
Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Main author</th>
<th>Summary of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1</td>
<td>2015-05-08</td>
<td>C. Rolfes (Fraunhofer AISEC)</td>
<td>Initial structure of document created</td>
</tr>
<tr>
<td>v0.2</td>
<td>2015-06-08</td>
<td>C. Rolfes (Fraunhofer AISEC)</td>
<td>Regression update and photos</td>
</tr>
<tr>
<td>v0.3</td>
<td>2015-06-10</td>
<td>C. Rolfes (Fraunhofer AISEC), Norbert Bissmeyer (Fraunhofer SIT)</td>
<td>Draft version for review</td>
</tr>
<tr>
<td>v0.4</td>
<td>2015-07-20</td>
<td>C. Rolfes (Fraunhofer AISEC)</td>
<td>update</td>
</tr>
<tr>
<td>v1.0</td>
<td>2015-07-31</td>
<td>F. Kargl (UT)</td>
<td>integrating comments and final revision</td>
</tr>
<tr>
<td>v1.1</td>
<td>2015-08-20</td>
<td>C. Rolfes (Fraunhofer AISEC), Martin Moser (Escrypt)</td>
<td>update of test result section with FOT PCB</td>
</tr>
</tbody>
</table>

**Approval**

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepared</td>
<td>C. Rolfes</td>
<td>2015-07-20</td>
</tr>
<tr>
<td>Reviewed</td>
<td>All Project Partners</td>
<td>2015-07-25</td>
</tr>
<tr>
<td>Authorized</td>
<td>F. Kargl</td>
<td>2015-07-31</td>
</tr>
</tbody>
</table>

**Circulation**

<table>
<thead>
<tr>
<th>Recipient</th>
<th>Date of submission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Partners</td>
<td>2015-08-20</td>
</tr>
<tr>
<td>European Commission</td>
<td>2015-08-20</td>
</tr>
</tbody>
</table>
# Contents

Glossary iv

1 Introduction 1
   1.1 Document Overview ........................................... 1

2 Verification and Test Environment 2
   2.1 Strategy .......................................................... 2
      2.1.1 RTL Simulation .............................................. 5
      2.1.2 FPGA Emulator .............................................. 5
      2.1.3 RTL-Simulation with ASIC netlist ......................... 6
      2.1.4 ASIC .......................................................... 6
   2.2 Environment ...................................................... 6
      2.2.1 Compiler ..................................................... 6

3 Target of Evaluation 7
   3.1 ASIC .............................................................. 7
      3.1.1 Hardware .................................................. 7
      3.1.2 Firmware .................................................. 7
   3.2 Prototype Board ............................................... 8
   3.3 FOT PCB .......................................................... 9
   3.4 Host Platform .................................................. 9
      3.4.1 Hardware .................................................. 10
      3.4.2 Software .................................................. 11

4 Verification and Test Results 13
   4.1 ASIC functional tests .......................................... 13
   4.2 PCB Interfaces ................................................ 13
   4.3 VSS Kit System ............................................... 15

Bibliography 18
List of Figures

3.1 PRESERVE HSM ASIC chip ................................. 8
3.2 Chip die with logo ........................................ 8
3.3 ASIC prototype board with opened chip mounting .......... 9
3.4 PRESERVE FOT ASIC v2.0 board .......................... 10
3.5 VSS KIT 2 verification setup .............................. 11
## Glossary

<table>
<thead>
<tr>
<th>Abbrev</th>
<th>Synonyms</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
<td>An API is a particular set of specifications that software programs can follow to communicate with each other.</td>
<td></td>
</tr>
<tr>
<td>AU</td>
<td>Application Unit</td>
<td>Hardware unit in an ITS station running the ITS applications</td>
<td></td>
</tr>
<tr>
<td>ASN.1</td>
<td>Abstract Syntax Notation One</td>
<td>ASN.1 is a standard and flexible notation that describes data structures for representing, encoding, transmitting, and decoding data.</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
<td>A CA is an entity that issues digital certificates.</td>
<td></td>
</tr>
<tr>
<td>CAM</td>
<td>Cooperative Awareness Message</td>
<td>CAMs are sent by vehicles multiple times a second (typically up to 10 Hz), they are broadcasted unencrypted over a single-hop and thus receivable by any receiver within range. They contain the vehicle’s current position and speed, along with information such as steering wheel orientation, brake state, and vehicle length and width.</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abbrev</td>
<td>Synonyms</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>DENM</td>
<td>DNM</td>
<td>Decentralized Environmental Notification Message</td>
<td>A DENM transmission is triggered by a cooperative road hazard warning application, providing information to other ITS stations about a specific driving environment event or traffic event. The ITS station that receives the DENM is able to provide appropriate HMI information to the end user, who makes use of these information or takes actions in its driving and traveling.</td>
</tr>
<tr>
<td>ECC</td>
<td></td>
<td>Elliptic Curve Cryptography</td>
<td>ECC is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.</td>
</tr>
<tr>
<td>ECMR</td>
<td></td>
<td>ECU configuration measurement register</td>
<td>Register that contains the current measurement value during runtime</td>
</tr>
<tr>
<td>ECR</td>
<td></td>
<td>ECU configuration register</td>
<td>Register used for secure boot and authenticated boot inside the HSM (similar to platform configuration register inside a TPM)</td>
</tr>
<tr>
<td>ECRR</td>
<td></td>
<td>ECU configuration reference register</td>
<td>Register that contains the reference value corresponding to the ECMR</td>
</tr>
<tr>
<td>ECU</td>
<td></td>
<td>Electronic Control Unit</td>
<td></td>
</tr>
<tr>
<td>FOT</td>
<td></td>
<td>Field Operational Test</td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td></td>
<td>Field programmable gate array</td>
<td></td>
</tr>
<tr>
<td>HSM</td>
<td></td>
<td>Hardware Security Module</td>
<td></td>
</tr>
<tr>
<td>HU</td>
<td></td>
<td>Head-Unit</td>
<td></td>
</tr>
<tr>
<td>I2V</td>
<td>I2C</td>
<td>Infrastructure-to-Vehicle</td>
<td>Communication between infrastructure components like roadside units and vehicles</td>
</tr>
<tr>
<td>I2I</td>
<td></td>
<td>Infrastructure-to-Infrastructure</td>
<td>Communication between multiple infrastructure components like roadside units</td>
</tr>
<tr>
<td>IDM</td>
<td></td>
<td>Identity &amp; Trust Management module</td>
<td>Module that handles the long-term certificate of the ITS station</td>
</tr>
<tr>
<td>Abbrev</td>
<td>Synonyms</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>ILP</td>
<td></td>
<td>Inter Layer Proxy</td>
<td>Component introduced by the SeVeCom project, that captures and allows modification of messages between different layers of a communication stack.</td>
</tr>
<tr>
<td>IDK</td>
<td>Module Authentication Key</td>
<td>Device Identity Key</td>
<td>The Device Identity Key is introduced by EVITA and is used for HSM identification. The IDK can also be certified by a manufacturer authentication key.</td>
</tr>
<tr>
<td>ITS</td>
<td></td>
<td>Intelligent Transportation Systems</td>
<td>Intelligent Transport Systems (ITS) are systems to support transportation of goods and humans with information and communication technologies in order to efficiently and safely use the transport infrastructure and transport means (cars, trains, planes, ships).</td>
</tr>
<tr>
<td>ITS-S</td>
<td></td>
<td>ITS Station</td>
<td>Generic term for any ITS station like vehicle station, roadside unit, ...</td>
</tr>
<tr>
<td>IDM</td>
<td></td>
<td>ID &amp; Trust Management Module</td>
<td>Module responsible for ID management originating from SeVeCom project.</td>
</tr>
<tr>
<td>LTC</td>
<td></td>
<td>Long-Term Certificate</td>
<td>PRESERVE realization of an ETSI Enrolment Credential. The long-term certificate authenticates a station within the PKI, e.g., for PC refill and may contain identification data and properties.</td>
</tr>
<tr>
<td>LTCA</td>
<td></td>
<td>Long-Term Certificate Authority</td>
<td>PRESERVE realization of an ETSI Enrollment Credential Authority that is part of the PKI and responsible for issuing long-term certificates.</td>
</tr>
<tr>
<td>MAC</td>
<td></td>
<td>Media Access Control</td>
<td>The MAC data communication protocol sub-layer is a sublayer of the Data Link Layer specified in the seven-layer OSI model.</td>
</tr>
<tr>
<td>Abbrev</td>
<td>Synonyms</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>OBD</td>
<td></td>
<td>On-Board Diagnosis</td>
<td>OBD is a generic term referring to a vehicle's self-diagnostic and reporting capability that can be used by a repair technician to access the vehicle's sub-systems.</td>
</tr>
<tr>
<td>OEM</td>
<td></td>
<td>Original Equipment Manufacturer</td>
<td>Refers to a generic car manufacturer</td>
</tr>
<tr>
<td>OBU</td>
<td>IVS</td>
<td>On-Board Unit</td>
<td>An OBU is part of the V2X communication system at an ITS station. In different implementations different devices are used (e.g. CCU and AU)</td>
</tr>
<tr>
<td>PC</td>
<td>Short Term Certificate</td>
<td>Pseudonym Certificate</td>
<td>A short term certificate authenticates stations in G5A communication and contains data reduced to a minimum.</td>
</tr>
<tr>
<td>PCA</td>
<td></td>
<td>Pseudonym Certificate Authority</td>
<td>Certificate authority entity in the PKI that issues pseudonym certificates</td>
</tr>
<tr>
<td>PIM</td>
<td></td>
<td>Platform Integrity Module</td>
<td>Module responsible for ensuring in-vehicle component integrity originating from EVITA project</td>
</tr>
<tr>
<td>PKI</td>
<td></td>
<td>Public Key Infrastructure</td>
<td>A PKI is a set of hardware, software, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates.</td>
</tr>
<tr>
<td>TPM</td>
<td></td>
<td>Trusted Platform Module</td>
<td>A TPM is both, the name of a published specification detailing a secure crypto-processor that can store cryptographic keys, as well as the general name of implementations of that specification, often called the “TPM chip” or “TPM Security Device”.</td>
</tr>
<tr>
<td>UTC</td>
<td></td>
<td>Coordinated Universal Time</td>
<td>UTC is the primary time standard by which the world regulates clocks and time.</td>
</tr>
<tr>
<td>V2I</td>
<td>C2I</td>
<td>Vehicle-to-Infrastructure</td>
<td>Direct vehicle to roadside infrastructure communication using a wireless local area network</td>
</tr>
<tr>
<td>V2V</td>
<td>C2C</td>
<td>Vehicle-to-Vehicle</td>
<td>Direct vehicle(s) to vehicle(s) communication using a wireless local area network</td>
</tr>
<tr>
<td>Abbrev</td>
<td>Synonyms</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>V2X</td>
<td>C2X</td>
<td>Vehicle-to-Vehicle (V2V) and/or Vehicle-to-Infrastructure (V2I)</td>
<td>Direct vehicle(s) to vehicle(s) or vehicle(s) to infrastructure communication using a wireless local area network</td>
</tr>
<tr>
<td>VSA</td>
<td></td>
<td>Vehicle Security Architecture</td>
<td>General outcome of PRESERVE work package 1</td>
</tr>
<tr>
<td>VSS</td>
<td></td>
<td>V2X Security Subsystem</td>
<td>Close-to-market implementation of the PRESERVE VSA that is the outcome of PRESERVE work package 2</td>
</tr>
</tbody>
</table>
1 Introduction

The PRESERVE Vehicle Security Subsystem (VSS), which is developed in the EU-research project PRESERVE (Preparing Secure Vehicle-to-X Communication Systems), has its focus on a secure communication between vehicles or infrastructure components (V2X communication). V2X communication is used for example to send automatic position notifications, emergency warnings or traffic information that help the drivers reaching their destination safer and faster. Obviously, this goal cannot be reached without sophisticated security measures protecting integrity, authenticity, and privacy within the V2X communication systems against malicious attacks.

This document reports on the integration of the different components of the VSS Kit 2 including the ASIC-based Hardware Security Module (HSM) and the results of the validation tests. It extends the VSS system design (Deliverable 2.2 [1]) and the ASIC-based VSS prototype (Deliverable 2.3 [2]) described in earlier WP2 documents. While majority of functional tests of the VSS Kit is already described in D2.3, this D2.4 focuses on the functional tests of the ASIC and its integration into the VSS Kit. More extensive testing is then conducted in tests in the testbed, which are described in Deliverable 3.2 and joint tests described in D3.3.

1.1 Document Overview

In Chapter 2 we give a short introduction to the architecture of the verification setup used during the development lifecycle of the ASIC. Directly thereafter, we give an overview of the different components of the VSS Kit 2 and their interfaces in Chapter 3. The document concludes with the test results of each component in Chapter 4. Furthermore, the problems identified during the system integration with the prototype board are listed, which results in an improved design of the final ASIC board.

Please note that version 1.1. is an update of the report and includes the results of the ASIC PCB version 2.
2 Verification and Test Environment

2.1 Strategy

The main objective of the tests strategy described in this deliverable is to write tests for the top level verification of the SoC. This tests should be platform independent and can be reused for the four different devices under test (DUT) that have been created during the development cycle of the ASIC.

- RTL-Simulation
- FPGA Emulation
- RTL-Simulation with ASIC timing netlist (Place & Route)
- ASIC
- VSS Kit 2 (benchmark)

The top level verification does not replace the module verification. Module verification has been conducted during module design and is described as part of D2.3. In fact, the top level verification relies on the assumption that all modules are verified in detail and work as expected. So, the purpose of the top level verification is to make some basic-level tests if the module is connected to the bus and responsive and to test more complex scenarios, like interaction with other modules.

The different tests are all written in C. The test suite consists of a lot of small tests, each with a different focus. Each test cases will always be run against defined testvector values and the result of the comparison between expected and obtained result will be reported as passed or failed (with an error code). It is also necessary that all tests can be started automatically. This is called a regression test. The goal of this approach is to be able to verify changes in the design immediately. If there is a new version of the top level design a new regression run will be started and we will get as a result what problems are solved and if new problems have arised, compared to the previous design.

The general approach of the ASIC functional test cases implementation is as follows:

- write the test in C (test.c)
- compile it (test.c -> test)

Simulation
  - convert the test in a ramfile (test -> test.srec)
– start the simulation testbench with the selected ramfile (test.srec)
– see the result (pass/failed) in the transcript

**FPGA/ASIC**

– load the binary with GRMON
– run the test
– see the result (pass/failed) at the UART interface

• collect the results and merge them to a report

Each test has a header which gives a short overview what steps will be carried out during this test case.

**Listing 2.1: Header**

```
//***********************************************************************
//
// Filename : reg_rw.c
// Version : 1.0
// Created : 14.02.2012
// Author : Martin Baer
// Organization : Fraunhofer AISEC – EMS
// Copyright (c) 2012, Martin Baer

//***********************************************************************
```

We have to include some files to get the test working.

**Listing 2.2: include files**

```
#include "register.h"
#include "global_functions.c"
```

For example, the registers of the SoC modules are defined in the register.h. This definition makes it easy for us to use the registers and port the test to different memory mappings.
In the main routine of the test, the steps described in the header are implemented. This is an example and so not all registers are listed, but only one in every step as an example.

Listing 2.4: main

```c
int main()
{
    //*****************************************************************************
    //
    // STEP 0: Initialize
    //
    glob_start_test();

    //*****************************************************************************
    //
    // STEP 1: Check resetvalues
    //
    if ((UART_DATA & 0xFFFFFF00) != 0x00000000)
        glob_save_error(10);

    //*****************************************************************************
    //
    // STEP 2: Check implvalues
    //
    UART_STAT = 0xFFFFFFFF;
    if (UART_STAT != 0x00000086)
        glob_save_error(22);

    //*****************************************************************************
    //
    // STEP 3: Check the results
    //
    glob_check_errors();
    return 0;
}
```
As you can see, in each step the result is checked against the expected values. If the check fails, an error code is saved. The function glob_check_errors is checking for errors, which arose during the execution of the test case. The result is then PASSED or FAILED. In the case of an error the error code is printed, too.

Although a top level verification is done, the tests were adopted to the different kind of modules. For every module we have the following test types:

- register read-write test - check if the module is connected.
- simple internal function tests - check if the module works as expected (only exemplary, main work should be done in module verification)
- if necessary, tests where the module interact with other modules (e.g. interrupt is triggered)
- test including external communication

Every test case is located in a separate folder, together with the command files, Makefiles, etc. Also, in this folder is a README file located. In this file the test is described in a defined structure and a more detailed description can be found in the header of the sourcecode. Hence, it possible to iterate through all test folders, readout the README files and create a testplan with this information automatically.

### 2.1.1 RTL Simulation

A first run of these tests was conducted in the RTL simulation. For internal functions of modules, this is straight-forward. Because we do not only want to run internal test, we expanded this testbench with external I/O models where the external ports/pins of the chip are connected to the complements in the testbench. UART, SPI and I2C were tested with the help of these models. Because the protocols of USB and Ethernet are more complex we were not able to do a simulation of the full stack and utilized an FPGA emulator for these tests.

### 2.1.2 FPGA Emulator

On the FPGA level the same tests are reused. Because there is no testbench around the FPGA we have to test the connections in a different way. Instead of a testbench we used a signal analyzer (like Bus Pirate) or directly connected the external interfaces to a host PC. Of course, this part of the verification was automated as a regression, too.
2.1.3 RTL-Simulation with ASIC netlist

This is the same scenario as the first RTL-Simulation. In addition, timing information of the target technology and information of the backend design process are included, e.g. provided as an SDF file. Therefore, this kind of simulation takes much longer than the previous pure top-level RTL simulation.

2.1.4 ASIC

We reused the emulator setup and replaced the FPGA with the ASIC prototype board for these tests that were conducted after shipment of the ASIC.

2.2 Environment

2.2.1 Compiler

To compile the C-testfiles a sparc-elf-compiler is needed. In our setup we used the sparc-elf-3.4.4 from the Gaisler homepage: http://www.gaisler.com/index.php/downloads
We build the tests with the following compiler flags:

-Wall enable all warnings
-O0 disable optimization, because some parts of the test programs may make no sense for the compiler and may be skipped therefore.
-I include some folders that are needed

We also have to create the srec file using the following command:

```
/usr/local/bin/sparc-elf-objcopy -O srec test test.srec
```
3 Target of Evaluation

This chapter describes the different system components tested in the verification. This includes the ASIC, firmware, prototype board, Nexcom box, and VSS kit software. The architecture, design, and implementation of the Vehicle Security Subsystem and its components are described in PRESERVE deliverables D1.2, D2.2 [1] and D2.3 [2] respectively.

3.1 ASIC

3.1.1 Hardware

The Hardware Security Module (HSM) is the main unit in charge of handling the security processing. It contains state-of-the-art crypto modules, which perform Elliptic Curve Cryptography (ECC) and AES encryption, incorporates a True Random Number Generator (TRNG) and a Physical Unclonable Function (PUF) module and is additionally able to calculate SHA-2 256-bit HASH values according to the NIST standard. These crypto IP cores are connected via an AMBA bus system to a SPARC V8 processor (Gaissler Leon 3) and are accessible via Ethernet, USB 2.0, SPI or I2C interfaces from the outside. Hence this design is a System on Chip (SoC) solution for high security applications with modern cryptography inside, manufactured in a 55nm process from UMC.

A detailed description of the ASIC can be found in Deliverables D2.2 [2] and D2.3 [2]. 92 packaged ASICs were available for testing. Figure 3.1 is showing photo of packaged chip. The PRESERVE logo is visible on the package and also the unpackaged chip die, see Figure 3.2.

3.1.2 Firmware

For the system integration test an adapted ASIC firmware (see D2.3 [2]) was used. It is stored in the flash of the board and copied to the external flash memory during booting. The firmware has an Ethernet driver for communication and an ECC driver for signature verification integrated. The tested firmware Version was asic_firmware v1.0.0
3.2 Prototype Board

Figure 3.3 shows the prototype board manufactured to conduct the ASIC functional tests and external interface test. It has a socket to easily exchange the ASIC under test. The schematics of the board are included in D2.3 [2]. The board has embedded an SRAM and flash chip for storing the firmware. Several interfaces (USB, Ethernet, Serial, JTAG), push buttons and LEDs can be used for interaction with the board and debugging.
3.3 FOT PCB

After successful functional verification of the ASIC and the prototype board 50 chips were selected to be soldered on the final PCBs, which can then be used for future testing and deployment. As you can see in Figure 3.4 the chips are mounted directly on the board instead of using the prototype socket. Apart from that, the external interfaces and the dimension are be the same as the prototype board.

As our testing revealed issues with the v1.0 boards and especially the external SRAM, which required the ASICs to be clocked at a lower rate, we also designed a revised version of the PCB (v2.0) with a different low-latency SRAM that allows us to run at full frequency.

3.4 Host Platform

The VSS Kit is optimized and extensively tested for the Nexcom Vehicle Telematics Computer and the Hitachi communication stack. This setup is used in many automotive research projects, which makes it the ideal platform for dissemination of the VSS Kit. During the project the VSS Kit software was also successfully tested on different CPU and host architectures like NEC Linkbird (Mips), Cohda Mk3 (ARM), Denso WSU (PowerPC). Figure 3.5 shows the prototype board and the Nexcom box used for the system integration test together with some of the chip samples during regression tests.
3.4 Host Platform

3.4.1 Hardware

The Nexcom VTC 6201 is an embedded platform designed especially for the V2X communication. The box is powered by an Intel Atom™ D510 1.66GHz processor and has 2GB DDR2 RAM. It has an integrated GPS module and supports Gigabit LAN and Mini PC-Card extensions.

Key Features

- Built-in Intel® Atom™ D510 Dual Core 1.66GHz processor
- 2 GB DDR2 RAM
- SATA 2.5" HDD
- Mini-PCIe socket (PCIe + USB) x 1 (for WLAN module)
- 4 x USB ports
- Wide range DC input from 8V-60V
- Power ignition on/off delay controlled by software
- Availability of GPS, GPRS/UMTS/HSDPA
3.4 Host Platform

Figure 3.5: VSS KIT 2 verification setup

- Multiple display connections: Dual VGA and LVDS
- Support 2 x RS-232 (COM1, COM3), 1 x RS-232/RS-485 (COM2)
- 1 x GPIO 4IN, 4OUT
- Fanless design with ruggedized aluminum chassis
- 260mm (W) x 176mm (D) x 50mm (H)

Note that this is a platform that we expect to clearly exceed the capabilities of day 1 on-board-units where cost-constraints will likely require less powerful CPUs, less RAM, etc..

3.4.2 Software

The Box is running a Linux operating system with the Hitachi communication stack. A detailed description how to install and configure the VSS Kit V2 (ASIC and OpenSSL version) can be found in Deliverable D4.3 [3] and in the PRESERVE Technical Report 13 [5].

To switch between the SW-only and ASIC-based version of the VSS only the PCOM configuration file needs to be changed.
3.4 Host Platform

1. ############ ASIC related configuration \n   its.use.asic = 1  \n2. its.asic.address = 192.168.0.108  \n3. its.asic.port = 7654  \n
4 Verification and Test Results

4.1 ASIC functional tests

The ASIC evaluation and the corresponding functional module tests as well as mitigation strategies for the detected issues are described in detail in [2], Chapter 3.3. In summary, the issues do not have an impact or any drawbacks for the intended use-cases of the PRESERVE ASIC.

4.2 PCB Interfaces

The ASIC PCB has following external interfaces to communicate with the Host box.

- USB
- Ethernet
- I2C
- SPI

The USB and Ethernet are the main communications interfaces, whereas the slow backup interfaces I2C and SPI can not use the full potential of the ASIC. As discovered during the ASIC tests, the USB core is not fully functional. Therefore, the USB interface can not be used. The second high speed interface is working fine, so the PCB will be connected to the host using the Ethernet interface. The I2C and SPI were tested successfully. The following listing is the output of a small test program to verify the Ethernet and ECC Signature verification.

```bash
sudo ./esc_drv_ahb_eth_host -i 192.168.0.108 -v -r 100
Using TCP/IP connection
2015-08-16 11:37:38 FUNCT: Enter
preserveAsicInitConnection()
2015-08-16 11:37:38 DATA : 192.168.0.108 7654
2015-08-16 11:37:38 FUNCT: Leave
preserveAsicInitConnection()
Connecting to server @ 192.168.0.108:7654
Signature verification test will be performed.
Test repetitions 1
```
```plaintext
2015-08-16 11:37:39 FUNCT: Enter preserveAsicVerifySignature()
2015-08-16 11:37:39 FUNCT: Enter preserveAsicComm()
2015-08-16 11:37:39 INFO : Data to send (138 bytes):
  00 01 00 20 6C DC B1 3B 2B 8B DA 5D 68 B2 B8 18
  9C 26 B4 DB 66 A4 74 E1 CF 0C EF F9 DD AA 4C 05
  6D 94 5D C0 00 20 75 21 6F 0D 64 0D 90 19 CB 41
  45 9D AD F9 F0 FC 35 3B A6 85 F7 12 5A 73 39 0A
  D2 CF 91 E8 9A 50 00 20 CC A3 FE 20 ED 3E 3A 18
  86 A8 E7 6D 3A 7B 61 10 74 EA 4C E0 FC 35 A8 2B
  12 AC BD 9F 04 9D 7B B7 00 20 F9 FC 44 A8 22 68
  06 8B D1 D7 90 67 4A 80 D3 BF E6 80 57 A8 5B 2E
  73 80 44 AD C9 D3 F0 50 0C C2
2015-08-16 11:37:39 INFO : Data received (36 bytes):
  00 02 00 20 1F 0C 81 B5 C7 7F EF C0 55 9A A8 66
  60 76 4A CA DC C0 41 50 95 A9 21 83 83 C9 5F 61
  6F 56 85 CC
2015-08-16 11:37:39 FUNCT: Leave preserveAsicComm()
2015-08-16 11:37:39 FUNCT: Leave preserveAsicVerifySignature()
```

...
4.3 VSS Kit System

The system integration test of ASIC into the full VSS Kit was also successful. The VSS Kit 2.20 setup on NEXCOM was already verified and tested before availability of the ASIC HSM. The installation and setup of the Nexcom box, the integration of the VSS into the Hitachi communication stack and the handling of the signature verification by the ASIC were carried out without issues. The internal testsuite of the VSS software as well as the communication with the PKI passed for the VSS Kit 2.

Below you find an snippet of the VSSKit logfile showing the successful signature generation and verification with the OpenSSL based version.

```
0280bf800202015388dec640c6e19e0100520000043b9dc1e2e6a0ae6b164
b29cc92a27433f5cc963746988900951983bc3693db7ea03c06e542787588
ca0a7f9fd5b4f148f4e8ae8ebf9b753a72b9f1814b4765a3802e0210b24030
1000250401000000b0115043983154c0b020300000aff364646d8a85
36f77d3c31774a2376961ace56a38843b263b8360f8f1e9277bbe5080db4
764a5a7607ac024e78be1a167986bf09a5338aadf70704d785000014de
9a231eb04038815d60524010b4c000019b9fa0a96849db44300100640f1
004c6e20e52c044ba1250555e4fb28962f76a27e7c5a6f8c7a3663c6808eb0
50544ef9127fd83011d35f6c2c33af83aca5000e5d52084d347173cf099
11:54:47.886006 SecureCommunicationModule :: treatSendingPDU : end
```

Test passed
Now the test is repeated with with the ASIC based version.

```
12:09:33.814176 LowLevelOpenSSLWithASIC :: LowLevelInit : preserveAsicInitConnection success
12:09:33.819952 SecureCommunicationModule :: treatSendingPDU : begin
12:09:33.820075 OutgoingMessageManager :: OutgoingMessageManager : aid of sender = 24
12:09:33.820640 OutgoingMessageManager :: CreateSignerInfoForCamProfile : digest will be used
12:09:33.820734 CryptoModule :: SignMessage : e6d134d4b18815d6
12:09:33.837269 LowLevelOpenSSL :: Sign : Success
12:09:33.837481 SecureCommunicationModule :: treatSendingPDU : 02158001e6d134d4b18815d60000014debcf097060524010a4c000019b9fa0a96849430100002ac1efb54696c4ba2f4a808d151c9ea533e722defe639686c081210762a8fc4fde8ca55b2d30de601d691423639711926ed40a21906b29363df9c169cd
12:09:33.837543 SecureCommunicationModule :: treatReceivedPDU : end
12:09:33.837583 SecureCommunicationModule :: treatReceivedPDU : begin
12:09:33.837683 SecureCommunicationModule :: treatReceivedPDU : 02158001e6d134d4b18815d60000014debcf097060524010a4c000019b9fa0a96849430100002ac1efb546966c4ba2f4a808d151c9ea533e722defe639686c081210762a8fc4fde8ca55b2d30de601d691423639711926ed40a21906b29363df9c169cd
```
Further tests were conducted as part of the internal FOT 2. A detailed analysis of the VSS Kit including benchmarks and attacker scenarios is published in PRESERVE D3.2: FOT Trial 2 Results [4] where also overall performance benchmarks are included.
Bibliography


