Horizon 2020 Program (2014-2020)

Cybersecurity, Trustworthy ICT Research & Innovation Actions
Security-by-design for end-to-end security
ICT 32-2014

Secure Hardware-Software Architectures for
Robust Computing Systems [†]

Deliverable D3.1: SHARCS System architectures and requirements

Abstract: This deliverable describes the system architectures of SHARCS applications and their hardware security requirements aligned with the application requirements, as analyzed in WP2. It further reports on the progress of the secure processor, secure memory and secure communication components.

Contractual Date of Delivery	Month 12
Actual Date of Delivery	Month 12
Deliverable Dissemination Level	Public (except of chapter 4 which is Confidential)
Editor	Ioannis Sourdis
Contributors	Chalmers, FORTH, OnApp, Neurasmus, Elektrobit
Quality Assurance	Dmitry Pidan, Manolis Stamatogiannakis

[†] The research leading to these results has received funding from the European Union Horizon 2020 Program (2014-2020) under grant agreement n° 644571.

The SHARCS Consortium

Foundation for Research and Technology – Hellas	Coordinator	Greece
Vrije Universiteit Amsterdam	Principal Contractor	The Netherlands
Chalmers Tekniska Högskola	Principal Contractor	Sweden
Technische Universität Braunschweig	Principal Contractor	Germany
Neurasmus BV	Principal Contractor	The Netherlands
OnApp Limited	Principal Contractor	United Kingdom
IBM - Science and Technology LTD	Principal Contractor	Israel
Elektrobit Automotive GmbH	Principal Contractor	Germany

Document Revisions & Quality Assurance

Internal Reviewers

1. Dmitry Pidan (IBM)
2. Manolis Stamatogiannakis (VU)

Revisions

Ver.	Date	By	Overview
1.0	21/12/2015	I. Sourdis (CHAL), E. Vasilakis (CHAL)	Finalizing deliverable.
0.8	18/12/2015	E. Vasilakis (CHAL), J. Thomson (OnApp), R. Seepers (NEUR), T. Kamm (EB), G. Christou (FORTH)	Addressing Quality Review comments.
0.7	8/12/2015	I. Sourdis (CHAL)	Deliverable ready for internal review.
0.6	5/12/2015	E. Vasilakis (CHAL), J. Thomson (OnApp), R. Seepers (NEUR), T. Kamm (EB), G. Christou (FORTH)	Comments addressed by partners.
0.5	18/11/2015	I. Sourdis (CHAL)	Introduction and Chalmers parts in Section 4 added. Comments provided in remaining parts.
0.4	15/11/2015	R. Seepers (NEUR), T. Kamm (EB), G. Christou (FORTH)	IMD and Automotive parts of Sections 2 and 3 added. Some parts of Section 4 completed.
0.3	14/10/2015	J. Thomson (OnApp)	Cloud description improved.
0.2	25/09/2015	J. Thomson (OnApp)	Modifications to cloud application.
0.1	07/09/2015	J. Thomson (OnApp)	Cloud hardware section preliminary.
0.0	24/6/2015	*Editor* I. Sourdis (CHAL)	Outline of the document created.

Contents

List of Figures

Acronyms

AES Advanced Encryption Standard. 37, 39

AES-NI Advanced Encryption Standard New Instructions. 12, 39

CAN Controlled Area Network. 18

CFI Control Flow Integrity. 29, 35–38

COTS Commercial Off-The-Shelf. 12, 21

CPU Central Processing Unit. 37, 39

DMA Direct Memory Access. 39

ECC Error Correcting Codes. 40

ECU Electronic Control Unit. 12, 15, 17–19, 28

FPGA Field Programmable Gate Array. 35, 43

GPU Graphics Processing Unit. 43

IMD Implantable Medical Device. 11, 12, 15, 16, 35–42

IPI Inter-Pulse Interval. 42

ISA Instruction Set Architecture. 36, 38

ISR Instruction Set Randomization. 29, 35–37

JOP Jump-oriented Programming. 37

LIN Local Interconnect Network. 18

MMU Memory Management Unit. 35, 39, 40

MPU Memory Protection Unit. 18

NIC Network Interface Card. 23

NIDS Network Intrusion Detection Systems. 12, 35, 43

OPEX Operating Expense. 23

RAM Random Access Memory. 39

ROM Read Only Memory. 35, 39, 41

ROP Return-oriented Programming. 37

SiMS Smart Implantable Medical System. 16

SISC Smart-Implant Security Core. 16, 36, 38

SoC System on Chip. 15–17, 35, 38–40

TCO Total Cost of Ownership. 23

TPM Trusted Platform Module. 12

1 Introduction

Workpackage 3 designs hardware security mechanisms for the SHARCS framework. As a first step towards this goal, in Task T3.1 we identified the particular security requirements to be supported in hardware for the SHARCS applications. This was performed based on the outcome of deliverable D2.1.

This deliverable (D3.1) is primarily the result of our activities in T3.1. It first describes the system architecture of each application (Implantable medical device, Automotive, Cloud computing) in Chapter 2. Subsequently, in Chapter 3 we identify the security requirements of each application that are to be supported in hardware and hence to be addressed in this workpackage[1]. Finally, the document describes in Chapter 4 our current progress and future plans for our activities in T3.2, T3.3 and T3.4.

In summary, the three SHARCS applications provide 3 very diverse systems as use cases considered in the project.

The implant application is a minimalistic system with two very basic microprocessors. It has very tight energy and area constraints and low performance requirements. Due to its required connection to external devices, the Implantable Medical Device (IMD) is vulnerable to various attacks. As explained later in Section 3.1, the main IMD security requirements that call for hardware support are the following:

- Protection against code injection and code reuse attacks;
- More efficient (faster, energy efficient) computing for encryption/decryption;
- Data integrity and data leakage protection;
- Dynamic key management.

[1] Unless there are available existing efficient solutions.

The automotive application is an embedded system which has as a basic building block an Electronic Control Unit (ECU) composed of an embedded processor, external memory and various peripherals and is supported by an embedded operating system. Such system has higher performance requirements but lower power consumption constraints compared to an IMD. Its security requirements that are to be supported in hardware, as discussed in Section 3.2, are primarily:

- Protection against code injection and code reuse attacks, and
- Data integrity, data leakage protection,

Additionally, improving the processing efficiency for cryptography, as well as dynamic key management may also be of interest.

Cloud computing is based on radically different infrastructure to either that of the IMD or Automotive applications. Clouds are hosted in large data centers that consist of one or multiple types of hardware platforms built mostly out of Commercial Off-The-Shelf (COTS) components. In general, cloud services provide a fraction of their computing resources to clients offering isolation and Quality of Service at the minimum possible ownership and maintenance (power, cooling, engineering) cost. Although it is extremely hard to introduce hardware changes in COTS platforms used in cloud computing, we have identified some cloud security requirements to best be addressed by hardware mechanisms, as explained in Section 3.3. In particular:

- Monitoring network traffic for suspicious or potentially damaging behaviour,
- Hardware accelerated encryption to aid end-user security and privacy, and
- Uniquely identifying components of a computing system and assure that they are running in an expected mode.

The first cloud requirement will be addressed in the project by a Network Intrusion Detection Systems (NIDS). For the rest of the cloud requirements, there are existing solutions, such as Intel's Advanced Encryption Standard New Instructions (AES-NI) and Trusted Platform Module (TPM); still we may explore new ones, which may have the potential to be integrated by hardware vendors in future platforms.

The remainder of the document expands on the above, describing the system considered for each one of the three applications (Chapter 2), their hardware security requirements (Chapter 3) and the current status of our

work in WP3 (Chapter 4). Finally, an overview of the SHARCS hardware mechanisms and their connection to the Application's hardware security requirements is offered in Appendix A.

2 System Architectures

This Chapter briefly describes the system architectures of the three SHARCS applications used in the project, namely the Implantable Medical Device (IMD), the automotive ECU and the cloud computing application. This description is the basis for subsequently identifying the the hardware security requirements that are to be addressed in the remaining activities of WP3.

2.1 System Architecture for an Implantable Medical Device

The SHARCS IMD Application is a novel, closed-loop, fully implantable neuromodulator that senses EEG and single-neuron recordings, detects seizures before they manifest, and prevents them through highly selective optogenetic (or electric) stimulation of cerebellar neurons. Although implant functionality is autonomous, the implant must also communicate with the outside world for overall control of the device (e.g. recalibration) as well as for sending patient-monitoring information to the patient, the doctor and so on.

The implant is smart, adaptive and autonomous, but it must also allow for operation under the remote control of an external handheld reader device, e.g. a smartphone. The handheld can perform various operations such as implant control, (re)calibration and data logging.

The neuromodulator is implemented in the Neurasmus System on Chip (SoC), depicted in Figure 2.1. Within SHARCS, we set the application boundary to contain all digital components plus wireless communication to/from the SoC. That is, we consider the following components:

1. **Sensor (digital):** This module obtains a digitized version of the sensory recording of an implanted electrode (saved in a register or memory location). It is the hardware block responsible for providing the

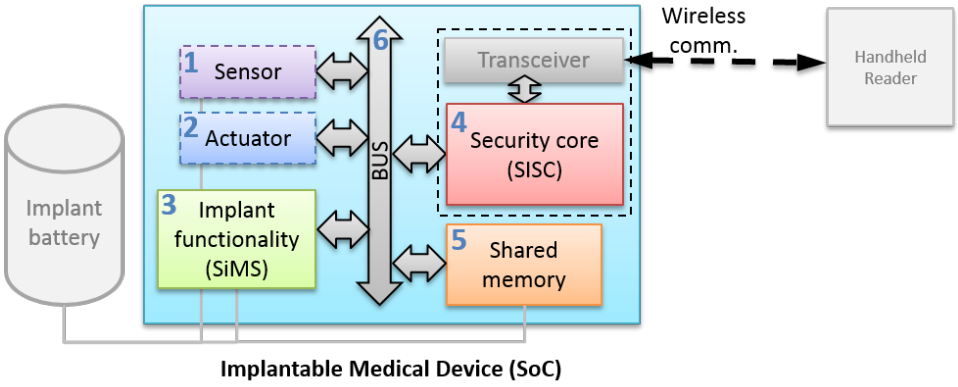

Figure 2.1: The Neurasmus-SoC which implements the implant application.

Smart Implantable Medical System (SiMS)-processor with correct and fresh data-samples at real-time every 10 ms. Within SHARCS, the digitized input to the sensor module will be emulated in hardware.

2. **Actuator (digital):** This hardware module forms the main output of the SoC. It is responsible for starting and stopping stimulation on the implanted electrode, based on a command received from the SiMS-processor. Within SHARCS, we include the memory location that the actuator reads as part of the application boundary.

3. **Smart Implantable Medical System (SiMS) processor (main implant functionality):** This module consists of three hardware components (the SiMS-processor with private instruction and data memory) and a software (the implant functionality) running on it. This module processes the input from the sensor and determines if the input shows seizure-related electrical activity. If so, it sends a message to the actuator module to start/stop stimulation.

4. **Smart-Implant Security Core (SISC) processor (secure communication):** The SISC processor is tasked with handling all (secured) communication to and from the SoC, without disrupting the main IMD functionality performed by the SiMS core. It has its own (private) instruction- and data-memory blocks.

5. **Shared-memory block:** This is a hardware component used for logging data originating from the SiMS (start and stop-time of seizure-activity) and SISC processor (data exchanged through the wireless communication).

6. **SoC interconnect (bus):** This hardware component is responsible for allowing communication between the components in the SoC.

A detailed description of the two main operations performed by the SoC, autonomous seizure prevention (main functionality) and external-device control, can be found in SHARCS deliverable D2.1.

2.2 System Architecture for Automotive applications

Modern premium cars have up to 80 Electronic Control Units (ECUs). Figure 2.2 shows a selection of different ECUs mounted in a car. Some ECUs, like the engine control and braking system, are essential in driving the car. Other systems such as the airbag are responsible for the safety of the driver. Window lift and Seat control are examples of ECUs controlling comfort functions, which are not necessarily needed to drive the car. The ECUs are interconnected by an on-board network. Furthermore a smart car may communicate with the outside world (e.g. Car-to-Car, Cloud service). Within SHARCS we would like to secure the complete system using a holistic approach.

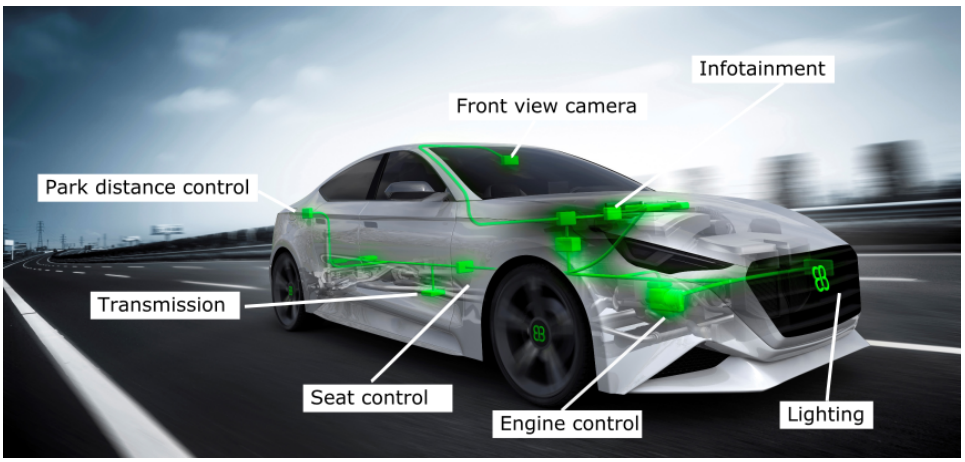

Figure 2.2: Automotive ECU examples.

2.2.1 General description of a typical ECU

An Electronic Control Unit (ECU) is an embedded system with a specific functionality in a car (e.g. engine control, brake). Figure 2.3 shows an abstract overview of an ECU. Depending on the ECU functionality there are different sensors(input) and actuators(output) connected. For example a Heating, Ventilation and Air Conditioning (HVAC) system needs a temperature sensor as an input. Depending on the measured and target temperature the system can control a heater. The ECUs are interconnected by different automotive buses. It is therefore important to secure all ECUs regardless

of their function. Otherwise it could be possible for a compromised ECU to gain access to others.

A typical automotive ECU consists, among others, of the following components:

- A processor core.
- Flash memory with a flash memory controller for persistent data storage.
- A Memory Protection Unit (MPU).
- A clock control unit.
- An interrupt controller handling external and internal interrupts.
- Controllers for access to communication networks (e.g. Controlled Area Network (CAN), Flexray, Local Interconnect Network (LIN), Automotive Ethernet).
- Sensor interface (e.g. Temperature, Speed, Camera, Radar).
- Interfaces for actuators (e.g. Actuators, Motors, Lamps, Relays).

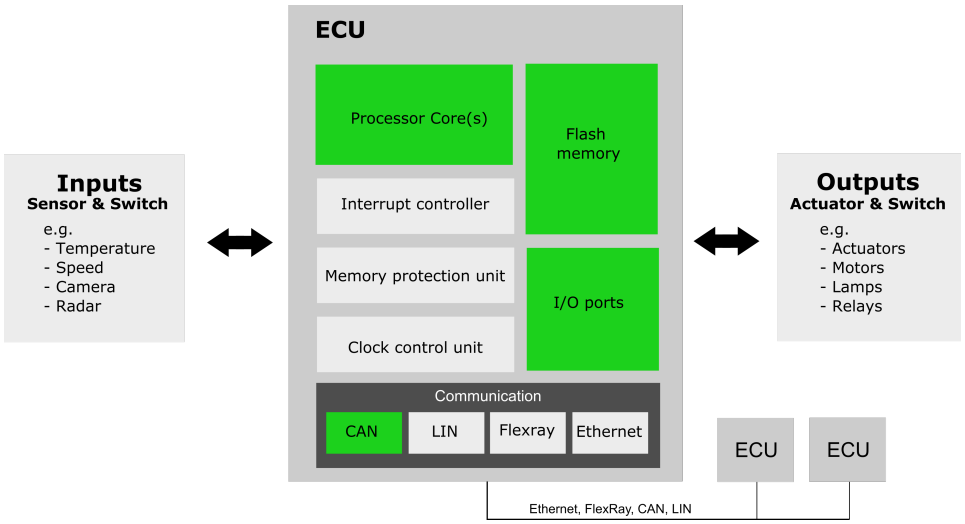

Figure 2.3: Typical ECU.

In an ECU usually a software application based on the AUTOSAR standard is executed on a specific automotive microcontroller(see Figure 2.4).

AUTOSAR[1] (AUTomotive Open System ARchitecture) is a worldwide development partnership of vehicle manufacturers, suppliers and other companies from the electronics, semiconductor and software industry. The layered architecture ensures the decoupling of the functionality from the supporting hardware and software services.

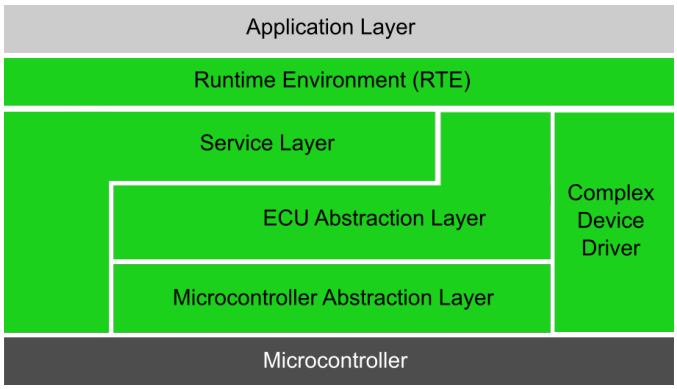

Figure 2.4: AUTOSAR software stack

2.2.2 General description of a typical car network

All ECUs are interconnected to an on-board network by different automotive busses like CAN, Flexray or Ethernet. The on-board network architecture is different between every car manufacturer and even car model. An exemplary network is shown in Figure 2.5. All ECUs are usually combined into groups like for example Body Electronics (e.g. Window Lift, Lighting), Infotainment (e.g. Head Unit, Instrument Cluster), Chassis and Safety (e.g. Electric Power Steering, Airbag) and Powertrain(e.g. Engine control, transmission). Communication between the different groups is possible over a gateway. It is therefore important to secure all ECUs as well as the communication between them regardless of their function. Otherwise it could be possible for a compromised ECU to gain access to others.

2.2.3 General description of a connected car

Modern cars are more and more connected to the outside world (Figure 2.6). They can communicate with other cars (Car-to-Car), with all kinds of infrastructure (Car-to-Infrastructure), with Cloud Services (e.g. real-time navigation or backup of settings) and with user appliances, such as smart-phones,

[1] http://www.autosar.org/

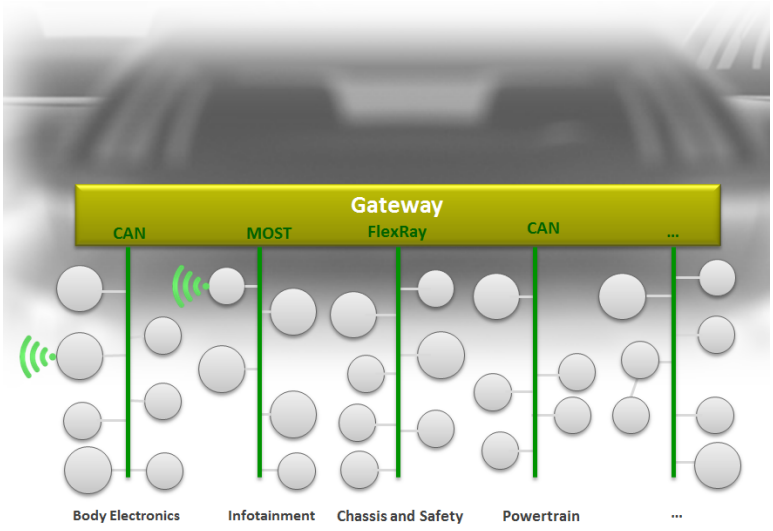

Figure 2.5: Automotive on-board network example

which can control it remotely. Because of the wireless connectivity a possible vulnerability can be remotely exploited in a large number of vehicles. This must be prevented by security mechanisms at all hazards.

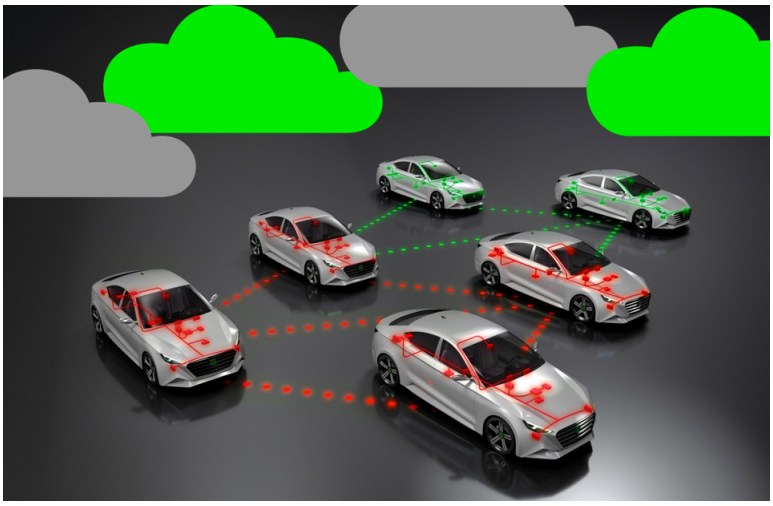

Figure 2.6: Connected car example

2.3 System Architecture for Cloud Computing

In this document we describe some of the background relevant to the hardware requirements of cloud application, while more of the detailed description is provided in D2.1. OnApp's customers are cloud service providers who install the software platform onto hardware that they have available in the data center. Sometimes they specifically purchase new hardware for deploying the platform but generally they use COTS legacy hardware that they have already in the data center. An advantage of the OnApp suite is that it is hardware and vendor neutral, which means that we try to support as many hardware devices as possible. That said, the current OnApp software stack is designed for x86-64 bit Intel hardware that is in widespread use in data centers. The hardware systems used for running OnApp also have to be able to support CentOS 5/6/7 Linux for the Control Panel and for supporting the Xen and KVM virtualisation platforms on the Hypervisor servers. The Backup server should also support CentOS Linux.

2.3.1 Background

Data centers are typically provisioned with an expected lifetime of 10-15 years but in that time there will likely be three to four server hardware refreshes[2]. Apart from the initial set-up/installation and maintenance the hardware will not change so the hardware effective life-time should be around 3-4 years.

We now describe, in brief, some of the hardware primitives and accelerator extensions that are built into hardware that is relevant to the Cloud application domain.

2.3.2 Trusted Platform Module (TPM)

The use of Trusted Platform Module (TPM) as part of the Trusted Execution Platform (TXT) is limited to motherboard and server vendors that have added a suitable TPM slot. Also, it is usually included as an extra for high-end server hardware. Some vendors provide the slot but do not provide the module, which for security reasons will need to be added and provisioned at the time of installation. TPM is normally combined with a software solution to help improving the trusted computing by using the secured memory regions for holding sealed keys that are used for encryption and decryption. Windows OS takes advantage of TPM for enabling BitLocker encryption of the OS itself and the data contained on the hard drive. TPM also offers the

[2] http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white%20papers/lifecycle%20costing%20for%20data%20center_determining%20the%20true%20cost%20of%20data%20centers%20cooling.pdf

ability to check that the configuration of the system is in a known 'good' state by recording parameters associated with a trusted boot and then at the point a configuration state is to be attested those recorded values future points in time. Additionally, TPM offers features that are of potential interest to , like remote attestation and sealing.

2.3.3 Instruction set extensions for security acceleration (e.g. AES-NI)

A hardware feature that is more prevalent than TPM are the Advanced Encryption Standard (AES-NI) extensions to the x86-64 Instruction Set Architecture (ISA). Server processors that have been manufactured by Intel since 2010 are likely to have support for these extensions. Similarly AMD have adopted the extension in 2011 and as such AMD server processors manufactured beyond 2011 will generally have support for AES-NI. AES-NI enabled processors will allow for accelerated encryption and decryption for processes that use AES ciphers and use the correct ISA extensions. When using Secure Shell (SSH), or using an SSL terminator or other functions that require AES, those processes will benefit from improved performance. For example, OpenSSL that is used by a large number of web hosters and other transport layer security processes such as IPSec can benefit from AES-NI. In addition to the AES-NI extensions that benefit AES ciphers, there are extensions that help to accelerate other ciphers including SHA-1, SHA-256 etc. that are expected to be included in some of the SkyLake Intel processors due in 2016.

In addition to the two security components discussed above there is also the possibility of using specific accelerator add-in cards. Recently, it has been a trend to use GPU on graphics cards to accelerate security related functions such as AES cipher encryption/decryption.

2.3.4 Add-in cards

Add-in cards, of which the most popular at the time of writing are PCI-Express add-in cards, are sometimes used by data centers and are usually used either as additional RAID cards or network interface hardware. These cards are usually bought at the time the hardware is purchased. It is extremely rare that a data center owner customises the hardware himself, preferring to use standard tested cards that come pre-installed and sometimes certified by the hardware vendor. The most popular and prevalent add-in card type available on Intel boards are PCI-Express slots. On most server boards there will be at least one PCI-Express slot. Another consideration though is that to reduce the chassis footprint these cards will utilise angled riser cards that change the orientation of the card. This also limits their maximum size. Exact constraints of the physical card size, the number of

PCI slots, PCI supported version and number of lanes is dependent on the type and configuration of the motherboard and chassis.

Add-in cards have not typically been used for adding security features other than for specific applications. Part of the reason for this is that add-in cards add to the complexity of the system, increase the cost (Total Cost of Ownership (TCO) and Operating Expense (OPEX)), increase the likelihood of a failure and require more maintenance. Another reason for avoiding add-in cards is that in the cloud environment, only hardware that is passed through the hypervisor to the virtual machines will be exposed, unless hardware pass-through is used or a virtualisation solution exists. Hardware passthrough on customer deployments is very rare and would mean that only a set of virtual machines that have been assigned access will be able to utilise the hardware. Some companies expose particular hardware features but this is normally for their bare-metal offerings. One such offering is IBM's Softlayer that integrates with the Intel TXT platform[3] for exposing security primitives to the bare-metal platform[4].

The server hardware is selectable by end users. OnApp has a hardware requirements recommendation page[5], however ultimately it is the client's responsibility to install the system on the hardware.

2.3.5 OnApp Cloud requirements

For a cloud platform to be useful the control panel needs two Network Interface Cards (NICs). One NIC connects to the Internet and another NIC is private and connects to the hypervisor network. Using the Preboot Execution Environment (PXE) boot specification, the OnApp cloudboot platform can be loaded into RAM on a hypervisor via a network card. This means that the hypervisor hardware needs to support PXE boot and have virtualisation extensions. The cloudboot image is packed into an image with some of the user-space applications being loaded into 272 MB supported by CRAMFS but it is usually suggested that a hypervisor should have at least 8GB of RAM. The filesystem is then be to RAM as well as the hypervisor software. The storage controller nodes are also loaded into RAM on the cloudboot hypervisors. The storage controllers are VM images that run the Integrated Storage platform software that utilises locally attached storage on the hypervisors avoiding the requirement for an external SAN. If using Integrated Storage then one additional NIC is needed on the hypervisor hardware for the SAN traffic. Additionally, the appliances normally have another network interface to allow customers to interact with their VMs, avoiding having to

[3] http://www.softlayer.com/intel-txt

[4] http://www.softlayer.com/press/ibm-and-intel-bring-new-security-features-to-the-cloud

[5] http://onapp.us.com/platform/requirements

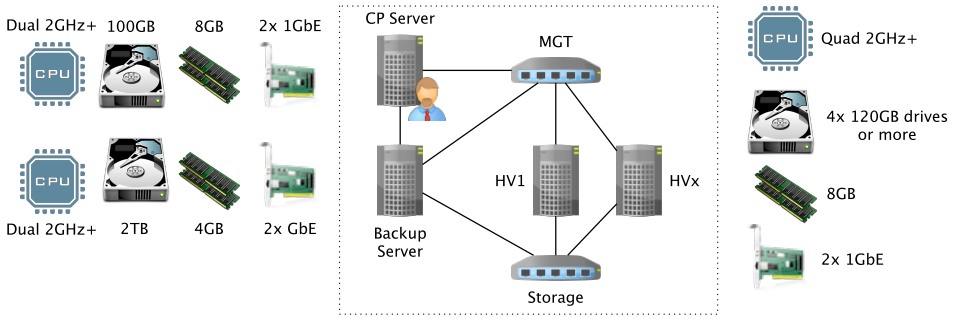

Figure 2.7: OnApp requirements and simplified N/W topology for a Cloud deployment using Integrated Storage.

pass traffic through the control panel or one of the other NIC interfaces. Figure 2.7 shows the hardware requirements in an illustrative manner.

2.3.6 Unified Extensible Firmware Interface (UEFI)

UEFI is a specification for software that acts as a bridge between the firmware and the OS level. Although there must be support in the hardware for UEFI the authors believe that the discussion of UEFI and secure boot best resides in the software discussion and as such will be captured in D4.1.

3 Hardware support for application requirements

This Chapter describes the security requirements of each SHARCS application that are to be supported in hardware. These requirements are extracted based on the outcome of deliverable 2.1 and the system description offered in the Chapter 2. In general, supporting security in hardware rather than software has numerous benefits, including:

- Power utilization. Due to having custom hardware accelerators that perform a limited subset of operations the power efficiency can be improved.
- Performance per unit cost. Usually the performance/cost point will be far greater than a system that has to perform generic computing to be attractive for people to buy the add-in boards.
- Other benefits of hardware security are: [1]

  Memory Access prevention

  Integrity assurance

  More secure against reverse engineering

  More resistance to power analysis attacks

  Ability to have long term key storage

  Independence from OS security

There are also disadvantages of using hardware instead of software that include:

- Long lead time for designing and synthesizing hardware;

[1] http://www.infosecwriters.com/text_resources/pdf/Security_Implications_of_HW_vs_SW_Cryptographic_Modules.pdf

- Inflexibility once the hardware has been developed to modify;
- Large non-recurring cost incurred;
- Cannot easily upgrade firmware if there is a security vulnerability detected;
- Very difficult for a non-established party to get hardware to market;
- Uptake will be much slower than for software.

3.1 Hardware Requirements for Implantable Medical Devices

The application analysis in SHARCS deliverable D2.1 has identified a number of IMD security requirements, which we repeat here for convenience:

SR-1.1 Security compliance with extra-functional constraints. Briefly stated, these extra-functional constraints are: 1) Real-time execution of the IMD functionality (10 ms); 2) Increase power and energy consumption by no more than 10%; 3) Increase device-area by no more than 30%.

SR-1.2 Security compliance with proper treatment delivery.

SR-1.3 Patient-data security and privacy.

SR-1.4 Patient safety & device accessibility.

SR-1.5 Security compliance with maintenance tasks.

Given the performance, power and energy requirements in SR-1.1, we opt to use hardware-based security techniques as these are typically more efficient (performance, energy) than software-based solutions. Considering the security threats specified in SHARCS D2.1, and the corresponding attacks that may realize them (analyzed in SHARCS D2.2), we require the hardware to perform the following functions:

HR-1.1 **Lightweight cryptography (SR-1.1)** Given security requirement SR-1.1, security should be added at minimal overheads to the IMD and not lead to a violation of its real-time deadline. The SISC core in the Neurasmus SoC is an ASIP processor targeting (generic) security primitives. In order to make SISC more efficient, we consider a variety of hardware-based enhancements to the SISC core as described in section 4.1.3.

HR-1.2 Prevent code injection and code reuse (SR-1.2 through SR-1.5): The SiMS and SISC core virtually have full control over the IMD and its data, and an attacker could violate security requirements SR-1.2, SR-1.4 and SR-1.5 if he could inspect or modify this code (through code injection or code reuse; see SHARCS D2.2 threat TH1.1-1). By applying instruction set randomization (see section 4.1.1), we limit the understanding an attacker has of the code and the possibilities for arbitrary code execution. Moreover, hardware-based control-flow integrity (section 4.1.2) is implemented to detect any unauthorized code modifications and, if such a modification is detected, implant functionality is reverted to a basic version stored on a fail-safe ROM (section 4.2.4).

HR-1.3 Prevent data manipulation (SR-1.3): As an attacker is not allowed to modify the private patient data stored on the IMD, we aim to guarantee data integrity through using error-detection/correcting codes on the memory (section 4.2.2).

HR-1.4 Prevent data leakage (SR-1.3): Patient data should not be accessible to an attacker, even in case of a security breach. As such, we opt for secure and encrypted IMD communication (section 4.3) enhanced by the SISC ASIP processor (described in section 4.1.3). Moreover, a memory management unit (section 4.2.3) is employed to restrict the memory locations available to each SoC component, such that compromising one component cannot affect the entire SoC.

HR-1.5 Enable emergency authentication (SR-1.4): Patient safety must be guaranteed even in case of emergencies. In the existing IMD-security protocol, key management relies on the initial (offline) distribution of security keys and identifiers, preventing unauthorized access during emergencies (as discussed in section 4.3). To strengthen the existing protocol, online key distribution is introduced (section 4.3) which may utilize custom hardware to minimize energy consumption (further facilitating SR-1.1).

3.2 Hardware Requirements for Automotive applications

The application evaluation in SHARCS deliverable D2.1 has identified a number of security requirements that apply to the Automotive Application, which we repeat here for convenience:

SR-2.1 Message manipulation: An attacker shall not be able to impersonate another sender of messages which are received by the controller on

a communication bus in such a way that the controller executes code which the attacker provides.

SR-2.2 **Data flash manipulation:** If there is a data flash module on the controller which can be manipulated by an attacker, the attacker shall not be able to manipulate the data flash in such a way that the controller executes code which the attacker provides.

SR-2.3 **Single controller execution:** If the attacker is able to manipulate a single controller in such a way that the controller executes code which the attacker provides, the method used for this controller shall not be possible on a different controller running the same software stack.

SR-2.4 **Software module isolation:** If an attacker is able to modify the source code of one of the basic software modules or the application modules, the attacker shall not be able to obtain information about data in the other basic software modules and application modules.

The Automotive Application is very security-critical and the on-board and off-board vehicle communication must be enhanced with security mechanisms to prevent attackers from manipulating functionality or gaining unauthorised access to those ECUs. Such protection is essential for the safety and security of passengers on the road. It is therefore important to have a hardware trust anchor, which is the first secure part of the ECU that all other security mechanisms relies on.

Furthermore many ECUs, like the braking system, are timing critical and any new introduced security mechanisms should only have a minimal performance impact. This can be achieved more efficiently by using hardware (as opposed to software) security features.

3.2.1 Hardware security requirements

HR-2.1 **Code injection (SR-2.2 through SR-2.3)** One of the main threats in the automotive application is the insertion of malicious code/data as well as the change of the program flow. This way, an attacker can either manipulate ECUs functionality, or gain an unauthorized access to the ECUs data. A compromised ECU for example can endanger the safety of vehicles or unlock optional software based functionality without paying for it. Hardware security mechanisms like Instruction Set Randomization(ISR) as well as Control Flow Integrity(CFI) might be appropriate to prevent these attacks and secure the system (see Section 4.1).

HR-2.2 **Code reuse (SR-2.2 through SR-2.3)** As vehicles are more and more connected (e.g. Car-to-Car, Cloud services) a possible vulnerability

can be remotely exploited in a large number of vehicles. Hardware security mechanisms like Instruction Set Randomization (ISR) as well as Control Flow Integrity (CFI) might be appropriate to prevent these attacks and secure the system (see Section 4.1).

HR-2.3 Data integrity, data leakage (SR-2.4) Smart cars are connected to the outside world (e.g. cloud services, Car-to-Car) and therefore more sensitive data must be protected against illegal access. Efficient encryption of shared memory data as described in section 4.2 might be useful to secure data.

Efficient encryption/decryption and dynamic key management as described in section4.3 could be used but are not required in the automotive use case. Several security mechanisms like Secure Hardware Extension(SHE) and secure onboard communication are already supported. Nevertheless, these newly developed hardware security features might be also interesting for automotive applications.

3.2.2 Hardware application requirements

- **Interoperability:** Developed hardware security mechanisms should be independent from the processor architecture (e.g. ARM, Tricore).
- **Performance:** The hardware security mechanisms should only have minimal impact on the execution time.
- **Power:** Power consumption overhead should be as little as possible but is not as critical as in the implant application.
- **Chip area:** The chip area overhead should be as small as possible to save costs.

3.3 Hardware Requirements for Cloud Application

The application evaluation in SHARCS deliverable D2.1 has identified a number of security requirements that apply to the Cloud Computing application, which we repeat here for convenience:

SR-3.1 End-user security and privacy

SR-3.2 Integrity of the platform and workloads

SR-3.3 Availability of the platform

SR-3.4 All operations must be attributable to a user

SR-3.5 All operations must be authenticated

In D2.1 the security threats related to the Cloud application domain have been enumerated. These threats formed the basis of security requirements that are captured in that same document. In D2.2 further investigation into the attack vectors and specific vulnerabilities was performed. Some of the risks can be mitigated through a combination of software and hardware techniques.

The power utilisation and energy efficiency for security solutions in the data center is not as critical as it is for the Automotive or IMD applications. The more sensitive constraint is that of performance of the hardware. Running security functions that are computationally expensive limits their utilisation and deployment in applications. Many of the requirements detailed in D2.1 could be satisfied by security techniques implemented in software for generic computing units or in hardware for dedicated units. As stated earlier there are a number of benefits of running security functions in hardware including better power utilisation, improved performance and more difficult to break security mechanisms.

3.3.1 Security requirements - hardware

HR-3.1 Hardware accelerated encryption to aid end-user security and privacy (SR-3.1): Data and operations within a VM that is owned by an end-user (user role: “End-user”) are private and should not be accessible by other end-users or other stakeholders unless they are authorised to do so. Encryption mechanisms make data hard to read without the correct key thus help to improve privacy and maintain confidentiality of the data. Hardware accelerators can be used to offload some of the computation-intensive stages of the encryption process away from the main compute unit.

HR-3.2 Monitor network traffic for suspicious or potentially damaging behaviour (SR-3.1 through SR-3.3): Given that the Cloud application relies on network traffic entering and leaving the hypervisors and cloud site it is important to monitor and react to potentially damaging activity. The combination of a monitoring system and reactive components can help improving the robustness of the platform against several attack vectors. Given the large amount of traffic that exists in a cloud site and the importance of reacting in a timely manner it may be beneficial to have a dedicated hardware monitoring system. In the case where traffic flows need to be monitored, evaluated and possibly reacted upon faster than line-rate hardware will be the only possible solution.

HR-3.3 Uniquely identify components of a computing system and assure that they are running in an expected mode (SR-3.4, SR-3.5): A computer in an end-to-end secure system should be identifiable and changes to its standard runtime operation be detectable and reported. It is important to assure the end-users that the integrity of the hardware and base software system has not been compromised as all higher-level security primitives rely on a trusted base.

There are existing mechanisms that already cover **HR-3.1** and **HR3.3**, such as AES-NI and TPM, respectively. Consequently in WP3, we will focus more on the **HR-3.2** Cloud hardware requirement. In the next paragraph, we describe some additional Cloud features, which may be interesting or beneficial to support in hardware.

3.3.2 Hardware application, beneficial features

- **Encrypted Memory and I/O:** On multi-tenant platforms any breach or escalation of privileges from a virtual machine residing on the system, could potentially expose resources from other end-users. Isolation mechanisms are expected to ensure separation but there may be flaws by which other memory regions and other shared resources are exposed. Encrypting memory regions helps to maintain confidentiality of data and protect against injection attacks originating from outside the trusted region. The presence of a secure Memory Management Unit (MMU) allows for protection against memory snooping and warnings for out of bounds memory access. Future Intel processors are expected to bring in Software Guard Extensions (SGX)[2] that protect memory regions in 'enclaves'. SGX features will come out in (some) SkyLake processors that offer an inverse sandbox model through 18 new processor instructions (page 203 of https://software.intel.com/sites/default/files/332680-002.pdf). The availability of this hardware feature is outside of the control of SHARCS but once available

it could be investigated to determine if it offers improved end-to-end security.

- **Hardware assisted isolation mechanisms:** To protect the confidentiality of end-users' data and ensure that only authorised users can make changes to particular memory regions it may be possible to offer hardware assisted isolation mechanisms that don't rely on encryption alone. ARM's TrustZone is one such example where devices on a secure bus have extra provisions for security features and look for a security bit that is orthogonal to any overlaying hypervisor platform. Only authorised accesses will allow the security bit to be enabled and allow access to the secure regions.

- **Additional security primitives available in new hardware platforms:** During the course of the SHARCS project there will likely be new security mechanisms and features that become available. Some of the Instruction Set Architecture modifications expected in future Intel platforms include accelerators for certain parts of cryptographic functions as well as other security enhancing functions[2]. If there is a security function that is available in hardware that is widely available, it makes sense to develop software that can make use of it. There is no difference in the Cloud case where functionality from the underlying hardware could be exposed via the virtualisation platform given sufficient incentive/benefit for doing so. These new hardware primitives that are available in common off the shelf (COTS) equipment will be assessed as they become available and SHARCS may take advantage of them.

3.3.3 Application hardware requirements

- Operations - hardware failure rate should be industry standard. Less failures are better.

- Operations - hardware lifetime should be targeted at typical hardware refresh rates for the data center. If in the form of a solid state add-in card the lifetime should approach that of typical RAID and network cards. One benefit of add-in cards is that they can be migrated to other machines after the host system has been end-of-life'd.

- Operations - serviceability. Hardware should be easy to service with the system remaining online while serviced, unless it is a core component that needs the system to be powered down (e.g. Motherboard, CPU, RAM, BIOS/UEFI, add-in card).

[2] https://software.intel.com/en-us/isa-extensions/

- Operations - energy /power. The energy demand should be in line with similar hardware already in place in existing systems.

- Operations - heat. Related to the energy consumed. The heat output should be self-regulating to keep it within operational limits and not require anything other than air cooling. The waste heat produced should be in line with similar hardware already in place in existing add-in cards.

- Size / socket - Should use a standard motherboard PCI Express slot[3] and conform with PCI card dimensions for existing RAID cards and hardware that normally goes in data center hardware. Physical constraints are limited to that of the standard of the slot / connector used. Aside from the standardised connector the physical size may be constrained by the orientation, air-flow and cooling, as well as the placement of other physical components on the motherboard so is system dependent.

- Operation - Hardware should work with Linux OS with a driver module that is ideally supported by the community.

- Operation - Hardware - to work with Windows will need WHQL certification.

- Design - the hardware should be self-contained and not rely on shared memory spaces.

- Interoperability - Hardware should be compatible with standard Intel Motherboard layout.

- Compatibility - should be compatible with legacy hardware.

- Business - cost. Hardware cost should be in line with market rates.

- Business - Should be 'free' (without licence restrictions) to use for commercial usage.

- Business - Performance acceleration per price unit should be greater than that of the cost of buying another system to be integrated.

- Performance - Using the hardware module, all sub-system performance should not be degraded by more than 5%

[3] https://pcisig.com/specifications/pciexpress/

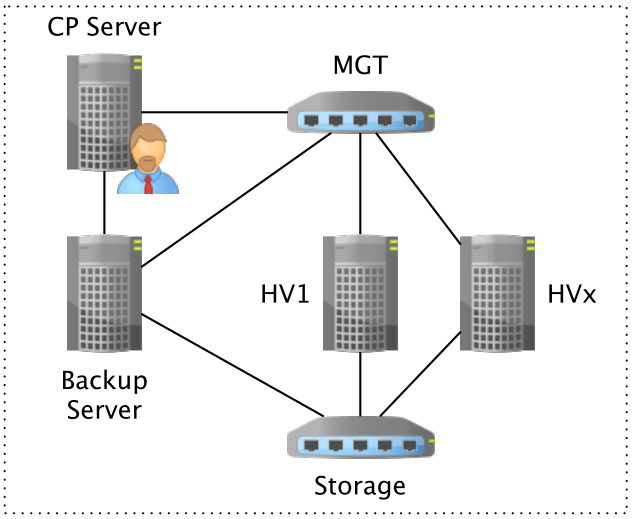

Figure 3.1: Diagram showing the computing systems involved in the cloud application.

3.3.4 Hardware assumptions

- Data maintained on the storage system does not suffer from corruption. Can assure this through multiple replicas and RAID style redundancy with error correction.
- Data stored in the memory subsystem is correct and free from errors due to ECC.
- Transmission of information in cables is error free. This allows for data correction at end-points that corrects non-malicious modification of the data due to erroneous transmission[4].

[4] https://www.fsl.cs.sunysb.edu/docs/integrity-storagess05/integrity.html

Bibliography

[1] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow integrity. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 340–353. ACM, 2005.

[2] I. Anati, S. Gueron, S. Johnson, and V. Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, page 10, 2013.

[3] T. Bletsch, X. Jiang, V. W. Freeh, and Z. Liang. Jump-oriented programming: a new class of code-reuse attack. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 30–40. ACM, 2011.

[4] ISO. Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms, ISO/IEC 9798-2:2008. International Standard, 2nd ed., 1999.

[5] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan. Trustlite: A security architecture for tiny embedded devices. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pages 10:1–10:14, New York, NY, USA, 2014. ACM.

[6] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens. Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 479–494, Berkeley, CA, USA, 2013. USENIX Association.

[7] R. Roemer, E. Buchanan, H. Shacham, and S. Savage. Return-oriented programming: Systems, languages, and applications. *ACM Transactions on Information and System Security (TISSEC)*, 15(1):2, 2012.

[8] R. M. Seepers, C. Strydis, P. Peris-Lopez, I. Sourdis, and C. I. De Zeeuw. Peak misdetection in heart-beat-based security: Characterization and tolerance. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 5401–5405. IEEE, 2014.

[9] R. M. Seepers, C. Strydis, I. Sourdis, and C. De Zeeuw. Enhancing heart-beat-based security for mhealth applications. *IEEE journal of Biomedical and Health Informatics (JBHI)*, –(-):1–9, 2015.

[10] R. M. Seepers, C. Strydis, I. Sourdis, and C. De Zeeuw. On using a von-neumann extractor heart-beat-based security. In *Security and Privacy in Computing and Communications (TrustCom), 2015 14th IEEE International Conference*, pages 1–8. IEEE, 2015.

[11] C. Strydis, R. M. Seepers, P. Peris-Lopez, D. Siskos, and I. Sourdis. A system architecture, processor, and communication protocol for secure implants. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(4):57, 2013.

[12] C. Strydis, D. Zhu, and G. N. Gaydadjiev. Profiling of symmetric-encryption algorithms for a novel biomedical-implant architecture. In *Proceedings of the 5th conference on Computing frontiers*, pages 231–240. ACM, 2008.

[13] C. Takano and Y. Ohta. Heart rate measurement based on a time-lapse image. *Medical engineering & physics*, 29(8):853–857, 2007.