Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

D3.3: Preliminary SPD Node Technologies Prototype Report

Due date of deliverable: M18 - 2013.02.28

Actual submission date: M20 - 2013.04.24

Start date of project: 01/09/2011 Duration: 36 months

Organisation name of lead contractor for this deliverable:

Integrated Systems Development, ISD

Revision [Issue 8]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Lorena de Celis	AT	18/02/2013	
David Abia	AT	18/02/2013	
Jacobo Domínguez	AT	18/02/2013	
Kyriakos Stefanidis	ATHENA	04/04/13	
Paolo Azzoni	ETH	15/03/13	
Stefano Gosetti	ETH	15/03/13	
George Dramitinos	ISD	16/04/13	
Viktor Do	SICS	30/01/13	
Christian Gehrmann	SICS	30/01/13	
Hans Thorsen	T2Data	30/01/13	
Chiara Peretti	TUC	05/03/13	
Kostas Rantos	TUC	05/03/13	
Georgios Hatzivasilis	TUC	05/03/13	
Kostas Fysarakis	TUC	05/03/13	
Alexandros Papanikolaou	TUC	05/03/13	
Dimitris Geneiatakis	TUC	05/03/13	
Harry Manifavas	TUC	07/02/13	
Luca Noli	UNIGE	07/02/13	
Paolo Gastaldo	UNIGE	07/02/13	
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Applicable Documents		
ID	**Document**	**Description**
[01]	TA	nSHIELD Technical Annex

Modification History		
Issue	**Date**	**Description**
Issue 1	06/12/12	First version of ToC.
Issue 2	04/02/13	Added contributions from several partners.
Issue 3	11/02/13	Added contributions from several partners.
Issue 4	18/02/13	Added contributions from several partners.
Issue 5	18/02/13	Added AT contribution
Issue 6	05/03/13	Added contributions from partners.
Issue 7	16/04/13	First full version of the deliverable.
Issue 8	23/04/13	Minor updates and corrections.

Executive Summary

This deliverable is focused on the detailed description of the node technologies under development in work package 3 that have reached a maturity level enabling their demonstration. These technologies will be made available to the application scenarios and can be used as building blocks for the project demonstrators. This deliverable will be updated and refined in the second part of the project based on the final requests received from the application scenarios and on the refined system architecture, metrics and composition strategy to be followed.

Contents

Figures

Tables

Glossary

Please refer to the Glossary document, which is common for all the deliverables in nSHIELD.

This Page is intentionally left blank

1 Introduction

The nSHIELD project proposes a layered architecture to provide intrinsic SPD features and functionalities to embedded systems. In this layered architecture work package 3 is responsible for the node layer that represents the lower level of the architecture, a basement constituted of real embedded devices on which the entire project will grow.

As already outlined in the TA, workpackage 3 aims to create an Intelligent ES HW/SW Platform that consists of three different kinds of Intelligent ES Nodes: nano node, micro/personal node and power node. These three categories of embedded systems will represent the basic components of the lower part of an SPD Pervasive System that will cover the possible requirements of several market areas: from field data acquisition, to transportation, to personal space, to home environment, to public infrastructures, etc.

This deliverable is focused on the publicly available description of the node technologies that are currently under development in work package 3, having reached to a maturity level that enables their demonstration. These technologies will be made available to the application scenarios and can be utilized as building blocks for the project demonstrators. This deliverable will be updated and refined in the second part of the project based on the final requests received from the application scenarios and on the refined system architecture, metrics and composition strategy to be followed.

The document is structured in the following sections:

1. Introduction: a brief introduction.
2. SDR/Cognitive Enabled node: SDR/Cognitive Enabled Node (CEN) technologies for generic application scenarios including technologies for secure booting, isolation of critical security tasks and power management.
3. Micro/Personal node: Technologies required by scenario 2 (Voice/Facial Recognition). It focuses mainly on biometric algorithms for SPD.
4. Power node: A horizontal technology for the GPU accelerated hashing and hash lookup is presented.
5. Dependable self-x Technologies: this section introduces horizontal SPD technologies that will be adopted at different levels, depending on the complexity of the node and considering its HW/SW capabilities, its requirements and its usage. The technologies are focused on areas such as denial-of-services and anonymity.
6. Cryptographic technologies: this section provides the assessment of horizontal SPD technologies focused specifically on hardware and software cryptography, on the use of crypto technologies to implement SPD embedded devices and prevent physical attacks at this level using defence crypto-based solutions.

2 SDR/Cognitive Enabled Node Technologies

The SDR/Cognitive enabled node technology prototypes available are the following:

- A hypervisor for ARM that allows security critical applications to run isolated, co-existing in the same system with less trustworthy or even insecure applications.
- A secure boot loader for ARM ensuring the integrity of the images to be loaded.
- Provision of confidentiality, integrity or/and authenticity services to nodes using smartcards.
- A family of Smart Power Units that is flexible enough to be used as a potential solution for power supply management a protection.

2.1 Hypervisor Prototype

As part of the nSHIELD prototyping efforts, SICS has developed a hypervisor that aims to enhance security in embedded systems by guaranteeing isolation and secure interaction between co-existing open software components and closed trusted security critical components. The target system architecture is depicted in the figure below.

We have been working with implementing an additional software layer, a hypervisor (the software managing the virtualization) at the most privileged level. In order to achieve this, the hypervisor utilizes the different operating modes of the CPU (privileged/unprivileged), the memory management unit (MMU) and the different domains to setup an access policy that satisfies the security of the embedded system. As the hypervisor runs in the most privileged mode to have full control over the hardware, that means in turn that all guest execution environments must be modified to run exclusively in user mode, otherwise the guest OS could potentially take over the system, destroying all security benefits of the hypervisor.

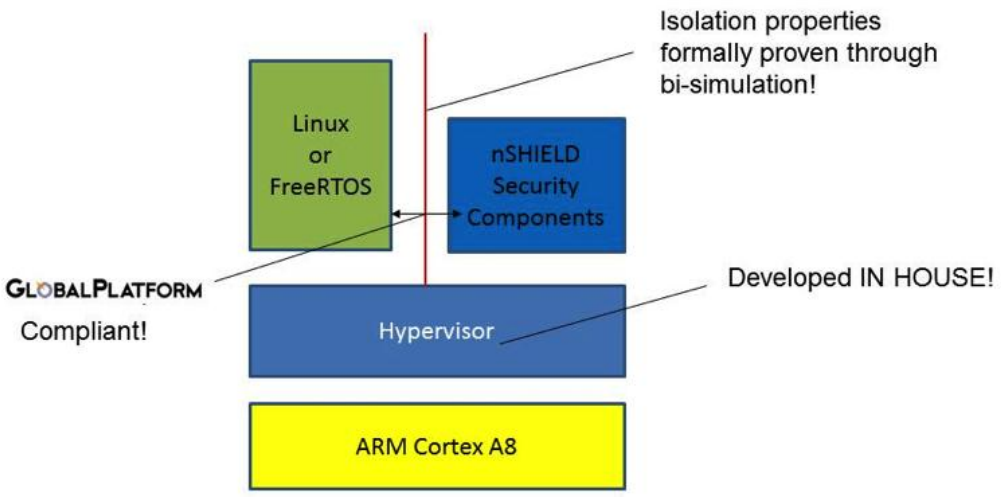


Figure 2-1: Target system.

Besides having the possibility to configure the access policies between different execution environments in real time, another major advantage is not having vast amounts of OS kernel code running in privileged mode. This makes the trusted computing base (TCB) imminently smaller, which also effectively minimizes the attack surface of the system. The overhead of a well-designed hypervisor is low enough that the performance trade away is well worth the increased security.

As the various nSHIELD nodes contain a number of complex and security sensitive components (especially those that handle cryptographic operations and keys), isolating these components from the rest of the system with the help of the hypervisor will be main priority. With the aim to provide a rich

environment for general-purpose applications to run, the end objective is to support a single general purpose OS like Linux together with other security critical applications, running in different execution environments, isolated from each other. The guest OS can use the services provided by the security critical applications, only through a well-defined interface that the hypervisor provides.

2.1.1 Platform Support

The hypervisor has mainly been developed in Open Virtual Platforms (OVP) [1], which is a tool to simulate virtual embedded systems with open source models. The simulation tool offers a fast and effective environment to test and develop software for different kinds of hardware and platforms, which eases the transition to real hardware.

The hypervisor currently supports the following:

CPU

- ARMv5 926EJ-S
- ARMv7 Cortex-A8

Platforms

- BeagleBoard-xM (Real hardware support)
- BeagleBone (Real hardware support)
- ST-Ericsson Nova Thor U8500 (Real hardware support)
- Integrator CP (OVP simulation)
- ARM Realview-eb (QEMU simulation)

The hypervisor has been developed with portability in mind; adding new platforms do not break or impact hypervisor functionality.

2.1.2 Beagleboard-xM

For the nSHIELD project, an agreement with the partners has been reached to use the BeagleBoard-xM [2] as the common platform as its identical to that used by Selex, the OMBRAv2 platform regarding CPU, DSP and other features, and both the SICS hypervisor and GNU SDR (software defined radio) [3] have been ported to it.

The BeagleBoard-XM uses the OMAP3530 system on chip, which includes an ARM Cortex-A8 single core CPU. Technical specification is given in the figure below.

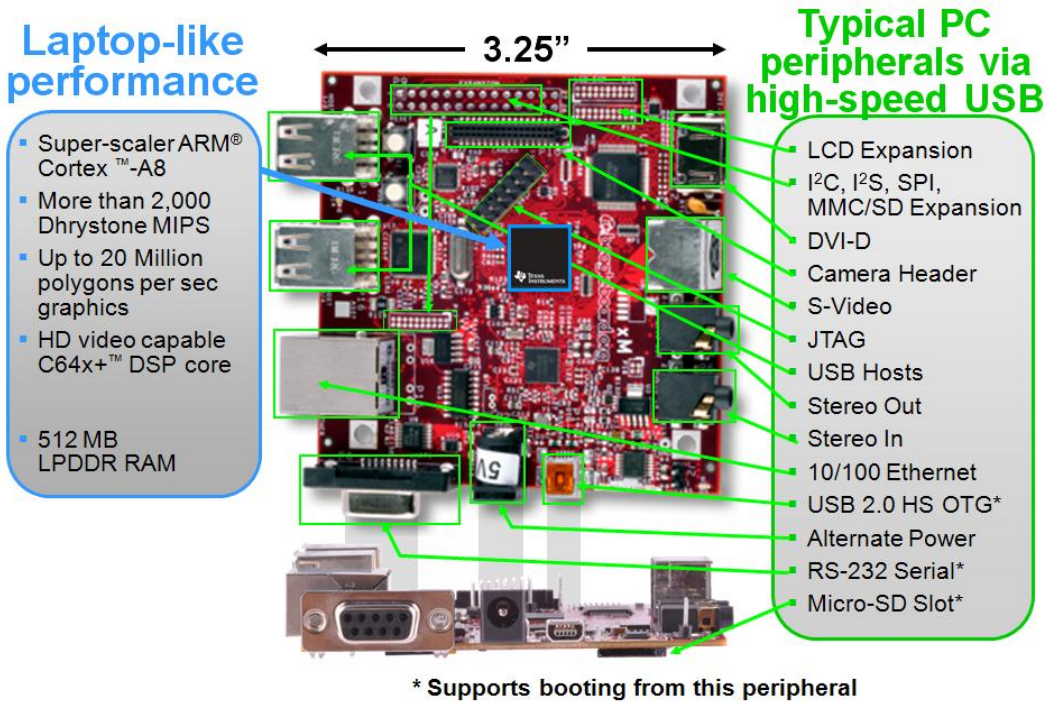


Figure 2-2: BeagleBoard platform

2.1.3 Secure Execution Environment Scenario for First Prototype

The prototype development is performed according to an iterative process where we gradually increase the hypervisor functionality and platform support according to the following rough development plan:

- FreeRTOS support, simulation and hardware support
- Linux support on OVP simulation
- Linux support on real hardware (BeagleBoard)
- Linux benchmark and hypervisor optimization

During the first year we have worked with providing a first prototype that show how the hypervisor can provide isolation that guarantees secure interaction between security critical applications and general user applications. The use case scenario that we have defined is the following:

- A general application asks a trusted service to perform security services such as cryptographic operations. As the security services are located in a different domain that the OS kernel and the regular application have no access to, it has to ask the hypervisor to perform these operations.
- In the same scenario, a malicious application has been introduced into the system trying to access information inside the trusted domain where the security applications reside. The hypervisor intervenes and stop the illegal access.

For the purpose of the first prototype, we have chosen FreeRTOS as our real time operating system.

2.1.4 FreeRTOS Port

The first prototype support FreeRTOS [4] as the real-time operating system. FreeRTOS is a very simple open source real time operating system with no file system or complex memory management. It has been paravirtualized to work in user mode on top of the SICS hypervisor together with its user applications.

In order to demonstrate the security benefits of the hypervisor, an application that mimics the nSHIELD security components was implemented. It contains security services that you would expect from a secure nSHIELD node, such as standard cryptographic operations with the private keys located inside its domain. The prototype application has been written in C language and essentially consists of three applications, the user application, the security application and the malicious application. The user and malicious application runs as user task on FreeRTOS while the security application runs independently outside the OS. They can be described as follows

- User application
    - Asking security application to see encrypted file
    - Asking security application to create signature of decrypted file
    - Asking security application to verify signature
- Malicious application
    - Try to access information inside trusted domain
- Trusted application
    - Providing encryption/decryption services (AES-128, RSA-1024)
    - Providing signature/verification services (SHA-256)

In the following chapters, we will show how the hypervisor provide a secure execution environment for the system.

2.1.4.1 Hypervisor Configuration

The hypervisor gives us the possibility to switch between different executions environments with their own memory configurations. We will here show the configuration that enforces an access policy that provides us with memory isolation between our hypervisor, OS kernel and our security critical applications. As FreeRTOS originally do not utilize the memory management unit, we are free to configure the system to satisfy our needs without major porting efforts. Through the linker script, we define where the different software regions are located in the memory. To keep the first prototype simple, each section uses 1MB of space, which is equivalent of one level 1 page table in ARM and have a 1:1 mapping from virtual to physical address. We have the following regions.

- **Hypervisor:** In this region we have the privileged hypervisor code and data, the vector table and the exception stacks. Dedicated memory addresses are also provided for the page tables and hardware peripheral devices.
- **Task:** The task region stores the wrapper codes for using the kernel hypercalls.
- **Kernel:** Stores the OS kernel code and data and the main function that starts up kernel tasks and the scheduler.
- **Trusted:** The security critical code resides in this memory region
- **Shared:** Stores library code and shared system resources
- **Taskpool:** Contains five regions that are used by the kernel for its tasks.
- **RPC:** Stores the RPC parameters.
- **Flash:** Stores flash data.

Now in order to allow full hardware control to the hypervisor, the boot file sets up the vector table and exception stacks, boots into the hypervisor in supervisor mode and sets up the page tables accordingly that satisfies our security policies. The hypervisor region will only be accessible in supervisor mode, and will generate a trap if any attempts to read or write the region while in user mode. Before the hypervisor hands over execution to the operating system, it makes sure that it switches the CPU state to user mode.

2.1.4.2 Page Table Domain and Access Permissions

Through the configuration in the page tables, each memory region in the previous section is assigned to one of the 16 domains available in the MMU. The assigned domain and access permissions for the page tables in the different memory regions can be seen in the table below. The hypervisor and device region address space are set to no access in user mode and the rest are set to read/write.

Table 2-1: Page table Access Permissions

Region	Domain	AP (User Mode)	AP (Supervisor mode)
Hypervisor	0	No access	R/W
Device	0	No access	R/W
Shared	0	R/W	R/W
Task	1	R/W	R/W
Kernel	2	R/W	R/W
Trusted	3	R/W	R/W
TaskPool	4	R/W	R/W
Shared RPC	5	R/W	R/W
Flash	6	R/W	R/W

2.1.4.3 CPU Domain Access Permission

In addition to the page table access permissions, the ARM CPU uses the CP15:c3 register to set access permissions to the different domains which is set to either client or no access. If the domain is set to client access, it means that it will check the access permission in the page table.

We have defined three virtual guest modes that the hypervisor can switch between and these are the following: kernel, task and the trusted mode. There is also a fourth guest mode interrupt, however it is only used by the hypervisor to handle interrupts and will not be shown here. By having different virtual guest modes, we can have different domain access configurations for each mode that suits our security needs. Regular applications are configured to run in the virtual guest mode task, while the OS kernel is configured to run in the virtual guest mode kernel. Most important, the trusted secure applications are configured to run in the virtual guest mode trusted. In our configurations, we have assigned a single domain that our trusted applications reside in (domain 3). It is however possible, to expand this with another trusted domain for other security critical applications to provide isolation between them. The hypervisor will then be responsible for switching between the different virtual guest modes and maintaining the virtual privilege level of the current mode. The table below shows how each virtual guest mode's memory configuration is set up.

Table 2-2: Domain access configuration for different guest modes

Domain	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RegionName							
Virtual Guestmode							
GM_Trusted	01	01	00	01	00	00	01
GM_Kernel	01	01	01	00	01	01	01
GM_Task	01	00	01	00	00	01	01

If we look at the domain access permission for the virtual guest mode *task* in the bottom row, the kernel memory area (domain 2) are set bit 00, which is no access. This is the mode, which applications run in, which isolates the kernel from the applications. At the virtual guest mode *kernel*, the domain access permission to hypervisor (domain 0), task (domain 1), kernel (domain 2) and task pool (domain 4) are all set to bit 01 which means client access. This means that for these domains, accesses are checked

against the access permission bit in the page table settings. Looking at the access permission in the corresponding table for user mode, the access permission for these domains is all set to read/write except for the hypervisor and device region. This protects the hypervisor software and the devices from illegal accesses when the processor operates in user mode.

As we can see in the configuration, the trusted domain (domain 3) is not accessible from the *task* or the *kernel* mode. Even if the task/kernel domain has been infected by a malicious application taking over the guest OS, it still cannot access the trusted domain. The only virtual guest mode that can access the trusted domain is *trusted* mode, which only the hypervisor can switch to. This way, a secure configuration is achieved by having our untrusted applications located in the task domain, while our trusted applications reside in the trusted domain.

One thing worth mentioning is that, not only do the different virtual guest modes have their own memory areas; they also have their own execution contexts. Whenever the hypervisor switches the virtual guest mode, it configures the domain access permission according to the configuration in the above table, saves and restores the context of the corresponding virtual guest mode.

To summarize this, each time a memory access is performed; the MMU looks at which domain the page table belongs to. The next step is to check the access permission for the domain. If it is set to no access, permission is denied. For client access, it continues to check the access permission in the page table. With the help of the MMU, page tables, domains and the different virtual guest modes, we have defined a secure access policy to our system.

2.1.4.4 Secure Services in Trusted Mode

Because the trusted domain is isolated and inaccessible from the other domains, the secure services running on the trusted domain are made available to the applications through dedicated hypercalls implemented in the hypervisor. This is called remote procedure call (RPC) and the arguments that are sent with the RPC tell what kind of services that it wants to perform. The RPC will execute a software interrupt instruction (SWI) which is a privileged operation causing it to trap to the hypervisor. The hypervisor SWI handler then analyse the parameters of the RPC and checks the configurations if the accesses are correct and allowed. The hypervisor then switch to virtual *trusted* mode performs the requested service and makes sure to only rely on encrypted and integrity protected data inside its trusted domain. When the trusted service is complete with the requested operations, it issues another hypercall "end RPC" which tells the hypervisor to return the computed information and yields back execution to the calling guest mode (the application that used the security service).

The hypervisor thus provides isolation between different execution environments and communication between the untrusted domains and the trusted domain are only allowed through secure interfaces on the trusted application.

2.1.5 Prototyping Results

The prototype runs FreeRTOS, its user applications and the security service applications on top of the hypervisor in the BeagleBoard-XM. In the setup, the BeagleBoard-XM is connected to a host computer through the RS-232 cable and is communicating through the UART port.

Through the output of the console, we can see how the user applications uses RPC hypercalls to request the hypervisor to execute the cryptographic operations on behalf of it inside the trusted domains. At the same time, the malicious application tries to access information from domains that it has no access to and is interfered by the hypervisor.

This shows how the hypervisor successfully manages to create a secure isolation between the different memory domains and as well provide a secure service in the trusted domain for the freeRTOS applications. This way, the security sensitive cryptographic keys stored in the trusted domain are never accessible by other applications; they can only ask the hypervisor to perform the operations on behalf of it.

The next step in verifying the correctness of the secure execution environment and isolation is having a third party code reviewing the hypervisor.

2.1.5.1 Benchmarks

In addition to the shown isolation properties of the hypervisor, we have benchmarked the hypervisor to show the performance overhead of virtualization. The tests covers the utilization of the kernel wrapper calls (user OS kernel calls that has to go through hypervisor), hypercalls (kernel systemcalls), interrupts and yielding which are the key performance burdens imposed by the hypervisor. All the tests use 5 tasks to perform a parallelizable math workload, which consists of 10000 work units. The task takes work units in a loop inside a critical section in where interrupts are disabled. It then leaves the critical section, which enables the interrupts and carries out the work unit. Each time a critical section is entered and exited, hypercalls are utilized. The tests are both run in preemptive and non-preemptive mode.

In the first test called "MathTest"', the tasks will continue to take work units until it is preempted by the FreeRTOS scheduler. One effect of running the test in non-preemptive mode is that one task will carry the whole workload by itself until it is finished and will never yield. In the second test "YieldingMathTest", the tasks yield after completing 5 work units, which will result in a significant execution of context switches. Whenever a task yields, it issues a hypercall to the hypervisor, which saves context and returns to the kernel handler function and in turn starts another task. In the third test "WrapperMathTest", the tasks call an arbitrary kernel function after completing 5 work units which results in a significant execution of the wrapper mechanism. When a kernel API function is called, hypercalls are issued to enter and exit virtual guest kernel mode, which modifies MMU settings.

2.1.5.2 Results

Looking at the comparison in performance for the baremetal kernel (no hypervisor) in the figure below, the performance overhead was at 0.4% for the benchmark MathTest. YieldingMathTest had an overhead at 11.1% while WrapperMathTest had an overhead of 7.9% which one can argue to be acceptable for the enhanced security tradeoff. It should be pointed out that these benchmarks demonstrate the exaggerated use of hypercalls, context switching and CPU mode change transitions.

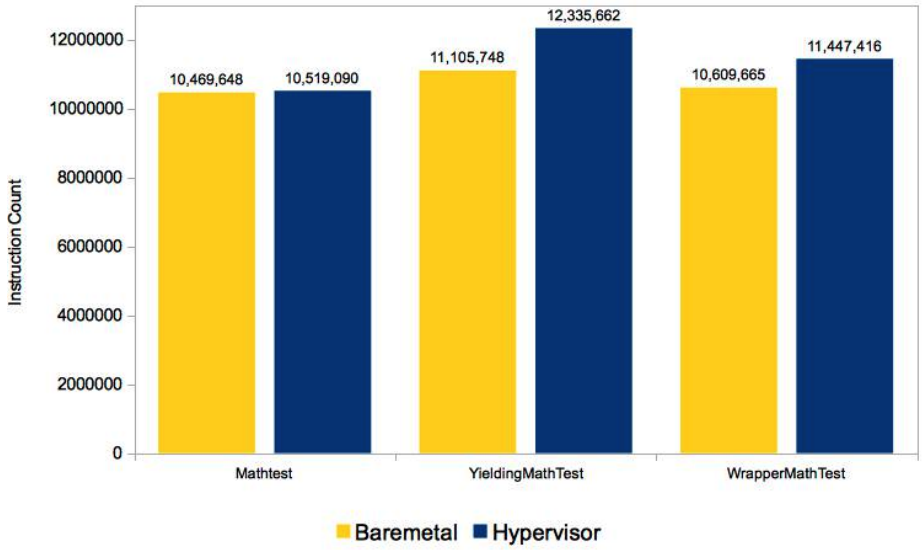


Figure 2-3: Benchmark performance of Hypervisor.

2.1.6 Next Step in Prototype Development - Linux Port

The next major step in the prototype development is porting Linux to work on top of the hypervisor. We are currently working with paravirtualizing the Linux Kernel, and adding new hypervisor functionality to support Linux. The current simulated platform for the Linux port is Integrator CP. Reason for not working

with the BeagleBoard on the Linux Port is mainly because there is no such model supported in the OVP simulation software. However, both platforms run the ARMv7 architecture CPU, which makes the eventual crossover to real hardware easier.

The complexity moving from a FreeRTOS port to a working Linux port is a large step, where the former OS has approximately 4k lines of code with no file systems, complex memory management, device drivers or networking to Linux with 15 million lines of code. Luckily, the Linux kernel has a huge user base, is open source and well documented.

2.1.7 Integrator CP Platform and Linux Kernel Build

The figure below shows the Integrator platform peripherals, available on the OVP site, which uses an ARMv7 Cortex-A9 uni-processor. In our case, we have switched the processor to a Cortex-A8 to closely match the BeagleBoard. However, both runs on the hypervisor even though we have not fully ported the Cortex-A9 to the hypervisor, due to their similarities. The Linux kernel used is version 2.6.34.3.

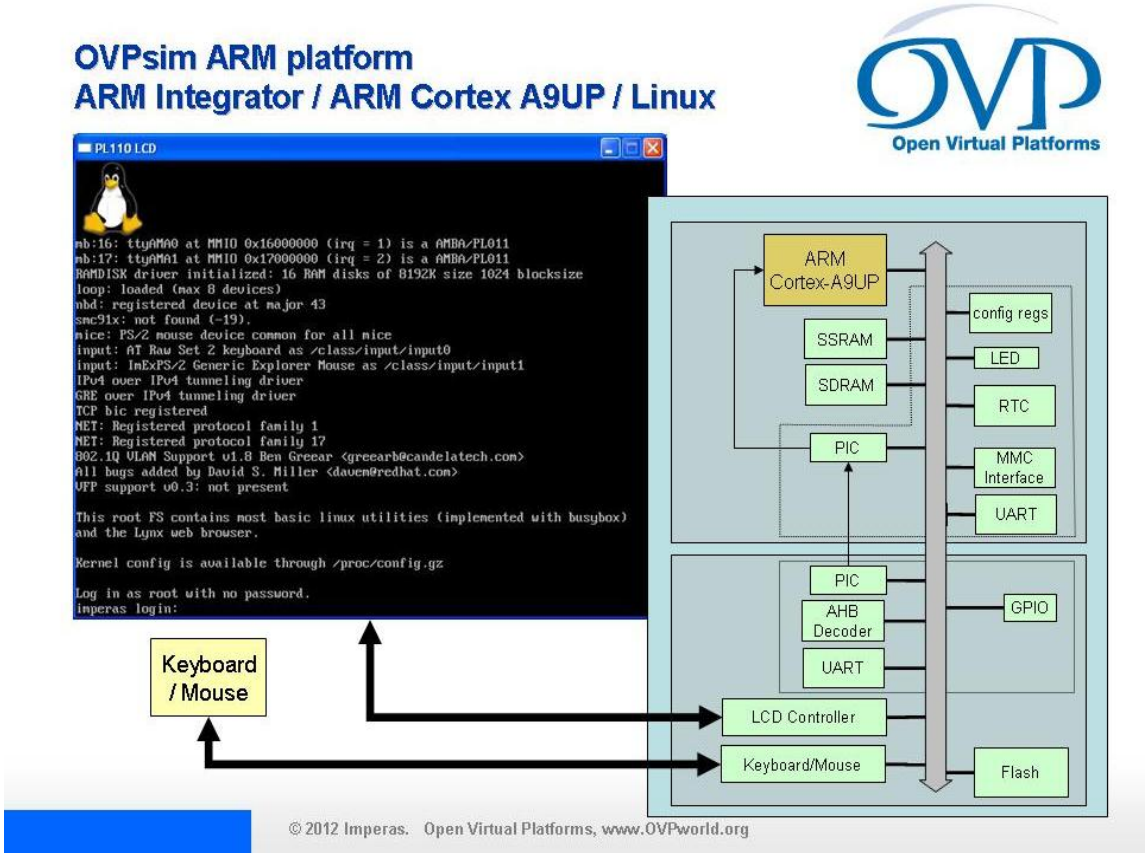


Figure 2-4: Integrator CP platform diagram.

The Integrator CP simulation platform from OVP gives us a good base for our Linux porting efforts in the nSHIELD project. To simplify things, the Linux 2.6.34.3 kernel has been modified and compiled with only UART support as a start. The kernel starts the bash shell command that communicates with a serial console through the UART. When we get a stable version of a paravirtualized kernel, we can start adding support for additional peripherals.

Linux kernel 2.6.34.3 build overview

- No graphic support
- Serial console (UART) communication with Linux bash shell

- Simple RAM-based filesystem (initramfs)
- No mouse/keyboard support (implement keyboard when Linux port successfully starts up bash shell)
- No SMP support (symmetric multiprocessing)
- Non preemptible kernel

2.1.8 Overview Linux Port

Currently, it's still too early to demonstrate the Linux porting work. The following is a list of what have been achieved, and what work is left.

Achieved

- Kernel 2 step boot process (image decompression and kernel bootup)
- Kernel platform initialization
- Exception handling connected through hypervisor
- Memory management foundation virtualized
- Platform IO virtualization (Timers, interrupt controller, UART)
- Hypercall / Systemcalls paravirtualized
- Scheduler started

TBD

- Running bash shell command line
- Generic device driver virtualization support
- Graphic, Keyboard and Mouse peripheral virtualization
- Linux DMA protection

2.2 Prototype Secure Firmware

2.2.1 Description

Cryptographic library with SHA1 hash and RSA decryption are integrated in the firmware. The signature of the image being loaded must have a corresponding signature.

2.2.2 Demonstration Scenarios

SICS will provide T2D with a hypervisor. The hypervisor will be signed by a command line utility prior to boot.

When the signature is verified execution of the hypervisor will start. The strength of the RSA key will be passed to the hypervisor as a SPD metric implemented as an extension to the ATAG structure.

Two systems in parallel will be used with different key lengths, 1024 and 2048 bits.

2.3 Power Management & Supply Protection Prototype

2.3.1 Description

Power management and supply protection is a fundamental feature to ensure the reliability of any electronic device. A general smart power unit, SPU, is shown below (see Figure 2-5).

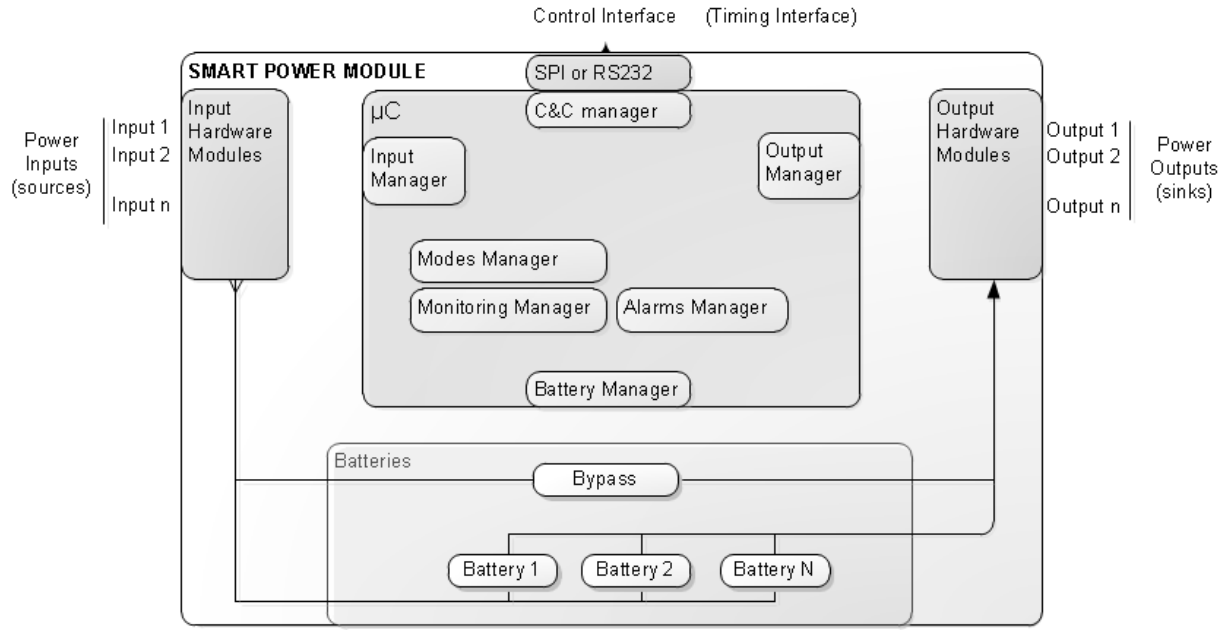


Figure 2-5: Smart Power Unit

It is planned to develop energy efficient and simple SPUs in order to be used by other nSHIELD modules. The final design and prototyping will depend on the requirements and needs of the nSHIELD modules that will use this power module.

2.3.2 Demonstration Scenarios

This module will be used in nSHIELD nodes in order to provide the required power management and supply protection. The scenario where this component will be used will be finalized when the nodes will be completed.

2.4 Smart Card Security Services in nSHIELD

2.4.1 Overview

A smartcard is a tamperproof secure device resilient to physical attacks used to perform secure transactions. Smartcards are used in a plethora of applications require security such as payment applications, healthcare, physical access control to mention a few. Smartcards can provide multiple security levels for sensitive data stored in them. For instance, a security key can be marked as read-only, while the read operation is accomplished only inside the smartcard. Even more the security key can be protected by a PIN to add one more security level. One of the main advantages of smart card solution is that all the sensitive operations are accomplished in the smart card rather than the terminal or application, which in many cases is not considered trustworthy. Smartcards among to others provide the following security services:

1. Message Authentication code
2. Encryption
3. Identity validity
4. Digital signatures
5. Hash functions
6. Secure key management

2.4.2 Communication with Smartcards

Smartcards have the structure depicted in the figure below.

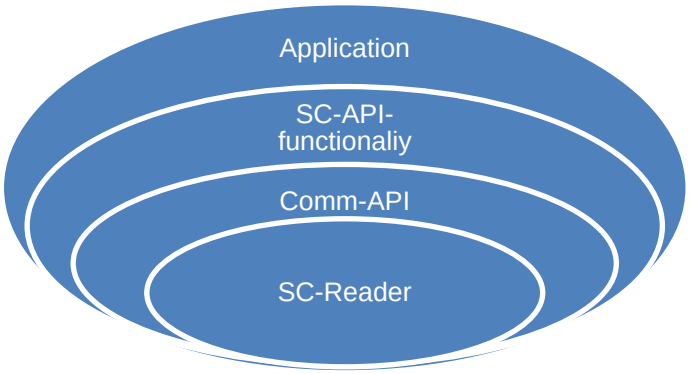


Figure 2-6: SmartCard communication structure

It should be noted that even in cases that smartcards do not provide a specific API for communication between the application and the smart card the communication with the can be accomplished by issuing direct command to the smartcard since the smartcards follows the ISO standards [5]. The general structure of a command in smartcards is illustrated in the table below.

Table 2-3: Smartcard request command format

Header				**Data**	
CLA	***INS***	***P1***	***P2***	***Length***	
Class where the command lies	*The command itself*	*Command first parameter*	*Command second parameter*	Data Length	Additional Data

The command can be issued towards the smartcard using the underlying communication of the terminal and the smartcard terminal (e.g. serial communication).

For every command issued toward to the smartcard there is a response which its format illustrated in the following table.

Table 2-4: Smart card response command format

Data	***Response Status***
The data returned by the smartcard	*Show the result of the requested command ,whether the command is successful or failed, and the reason of failure*

2.4.3 Smartcard File System and Data "Storage"

Smartcards file system structure is similar to those used in operating system. Particularly the ISO-7816 part 4 defines the structure of the file system as illustrated in the following figure. The master file (MF) can be considered as the root directory, while the dedicated and elementary files are the directories and the data file, in UNIX like operating system, correspondingly.

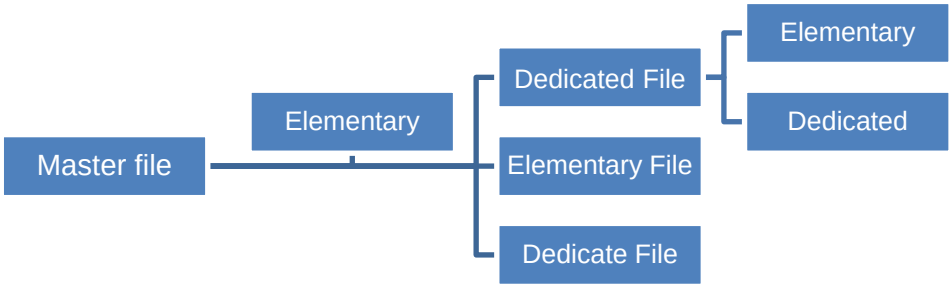


Figure 2-7: The logical structure of file system in Smartcards

In smartcards different kind of data can be stored either dynamically or statically, though their capacity is limited. For example, users' data or cryptographic keys for secure transactions can be stored. The header in data files defines also the access control rights. Every directory creates a security domain inheriting the security policy of its parent. The files in the smartcard can be protected with multiple ways:

- Different PIN
- Message authentication code
- Access control restrictions (read, write permissions)
- Digital signatures

This depends on the features incorporated in the smartcard.

2.4.4 Secure services with smart cards

Depending on the type and the manufacturer the smartcards support a number of cryptographic features, including:

- On-card generation of symmetric keys and public key algorithms key pairs
- Digital signatures (based on public key algorithms)
- Symmetric encryption and decryption
- External authentication (host to card)
- Internal authentication (card to host)
- Message authentication code
- Hash functions

Further, smartcards enable protected mode for highly sensitive data, which requires commands to be authenticated and integrity protected either with symmetric or asymmetric keys.

2.4.5 Using smartcards for security services: authentication example in the context of nSHIELD

For instance, consider the case where the overlay should authenticate a Micro-Node that incorporates a smartcard module. In that case the overlay generates a challenge and sends it to the micro node. The Micro-Node passes the challenge to the smartcard and requests it to create a message authentication code (MAC), assuming that we rely on symmetric key cryptography. The smartcard generates the MAC and sends it back to the Micro-Node that forwards the result to the overlay. The overlay can validate the received MAC either using a TPM or a software based security service. This procedure is illustrated in high level in the figure below. Note that the symmetric keys required by the Micro-Node can be either pre-installed in the smartcard or be generated dynamically in the smartcard itself.

.

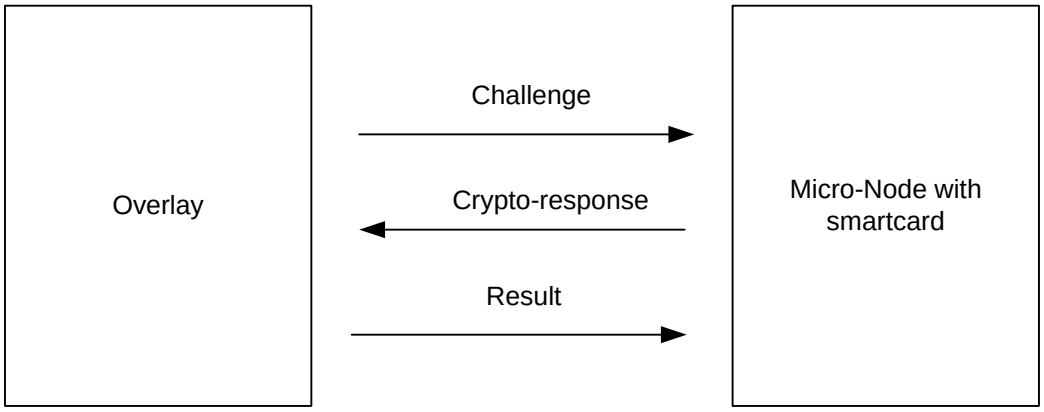


Figure 2-8: Example of authentication using smartcards. The overlay authenticates a Micro-Node

A similar procedure can be followed in the case where Micro-Node needs to authenticate the overlay. Particularly, the Micro-Node requests the smartcard to generate a random number which is forwarded to the overlay. The overlay generates the corresponding MAC and sends it back to the Micro-Node which requests the smart card to validate the generated MAC. Depending on the result it creates either success or failure response that is sent to the Micro-Node. Note that the smart card may not be able to validate itself the MAC. In that case, the smartcard will generate the MAC using the same challenge and the final validation will be accomplished by the Micro-Node by comparing the MACs received by the overlay and the smartcard. This procedure is depicted in the figure below.

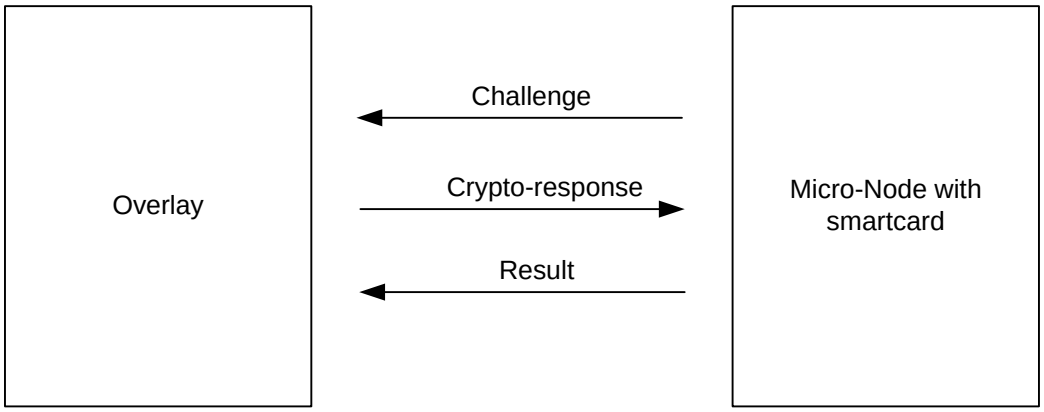


Figure 2-9: Example of authentication using smartcards. The Micro-Node authenticates the overlay.

3 Micro/Personal Node

The micro/personal ode technology prototype available is a face recognition system for people identification.

3.1 Face Recognition for People Identification

This section describes the prototype of the face recognition embedded system for people identification. The prototype is a demonstrator of the functionalities and potentialities of the system and represents a proof of concept of the technologies adopted for the face recognition that will be used to develop the final prototype. This prototype belong to the “Face and voice recognition” application scenario and has been developed during the first part of nSHIELD project. The final version of this prototype and the voice verification prototype, that together constitute the “Face and voice recognition” application scenario, will be developed in the second part of the project.

3.1.1 The Face Recognition Prototype

In this deliverable we illustrate an approach for face recognition based on the Eigenface method that satisfies the requirements of an SPD nSHIELD embedded system for face recognition. This method is based on the idea of extracting the basic features of the face: the objective is to reduce the problem to a lower dimension maintaining, at the same time, the level of dependability required for such an application context. This approach has been theoretically studied during the nineties and has been recently reconsidered because it provides a good ratio between the required resources and the quality of the results and it is well dimensioned for embedded systems. Today, it is becoming the most credited method for face recognition.

3.1.1.1 The Face Recognition Process

The core of this solution is the extraction of the principal components of the faces distribution, which is performed using the Principal Component Analysis (PCA) method. This method is also known in the pattern recognition context as Karhunen-Loève (KL) transform. The principal components of the faces are eigenvectors and can be computed from the covariance matrix of the face pictures set (faces to recognize). Every single eigenvector represents the feature set of the differences among the face picture set. The graphical representations of the eigenvectors are also similar to real faces (also called eigenfaces). The PCA method is autonomous and therefore it is particularly suggested for unsupervised and automatic face recognition systems.

The face recognition software has been developed using language C++, with C, C++, C#, Visual Basic API for Microsoft platform.

The recognition process that has been implemented is based on the following steps (see figure Figure 3-1):

- face detection,
- face normalization,
- face extraction,
- template creation,
- template matching,
- identification.

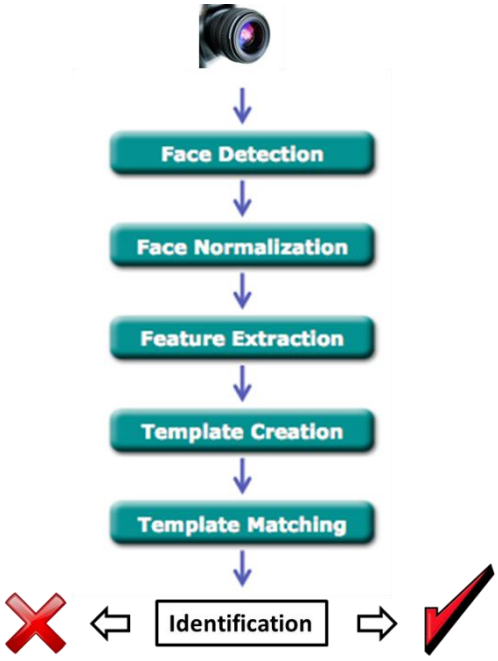


Figure 3-1: The identification process.

The face recognition software is composed of a set of software modules, which implement each of the logical steps of the face recognition process:

- Face Detection module – This module finds the position and the size of each visible face present in the acquired image. To reach this important goal the module contains a special algorithm for high speed face localization, which is based on the training of a chain of Haar features, acting as weak recognizers [6] and [7]. The training procedure has been implemented using a proprietary implementation of the algorithm called AdaBoost.
- Face Normalization module – This module identifies the features of morphological interest throughout the face area (eyes, mouth, and eyebrows), in order to properly rotate and scale the image. This operation moves all such points to pre-defined locations. As in the previous module, we developed an algorithm for localization of characteristic traits by training a chain of weak recognizers, [6] and [7]. This module has been designed privileging the requirement of operational efficiency, in order to be easily portable across embedded hardware and to limit the impact on PC performance that in the classic surveillance solutions are in charge of monitoring the people (i.e., continuous monitoring of the identity of an operator of a workstation). It is under development an evolution of the face localization, based on an algorithm belonging to the family of algorithms called Statistical Models of Appearance, which will allow the alignment of the face even in the presence of severe occlusions of some characteristic points (sunglasses, scarves etc.) [8].
- Feature Extraction module – This module selects the relevant features from the normalized image in order to maximize the robustness against environmental disturbance and noise, non-optimal pose, non-neutral facial expressions and variable illumination conditions. In the implementation, in order to maintain high quality results in presence of non-optimal environmental conditions, we adopted two specific solutions: the advanced techniques for illumination compensation [9] and the appropriate nonlinear differential filtering [10] that create features inherently insensitive to global illumination and not very sensitive to inaccuracies of alignment. Using a configuration file in XML, we can configure one or more parallel chains that are in charge of processing the biometric fingerprint.
- Biometric Template Creation module – The biometric template is implemented by a feature vector template that is created by this module. The extracted feature vector template is processed by a trained statistical engine and is reduced to a smaller one that optimally describes the user's identity.

- Biometric Template Matching module – The face recognition and the face verification processes are based on measurements of differences between biometric templates. The normalized template distance is the similarity value of the compared identities. This module is responsible for the creation of the normalized template distance.

Finally, a classification module has been developed in order to provide the previous modules with a horizontal tool capable to offer the classification features required during the various steps of the recognition process. The classification module is based on methods for statistical classification of models and is capable to manage information partially compromised by noise or occlusions. In particular, we implemented an optimized version of the Bayesian classifier [11], because the problem of biometric recognition is inherently a problem of classification between two classes only.

This module is used during the face detection, the relevant face features identification and during the final template comparison step.

As already anticipated, the classification module is based on statistical pattern analysis that treats the classification problem as follows:

Requirements:

- Inexpensive training process:
  - Rapid training.
  - Inexpensive hardware for training (e.g. workstation vs super-computer).
- Accurate classification.
- Fast classification process.

Classification problem:

- Given a tagged sample database (positives & negatives).
  - Select the optimal Features Extractors.
  - Train a Statistical Engine (classification algorithm).
- Goal: obtain a correct classification on unknown samples (generalization).

Starting from these concepts, the classification module has been developed using a recursive approach based on training and progressive adaptation. The architecture of the module is described in the following block diagram.

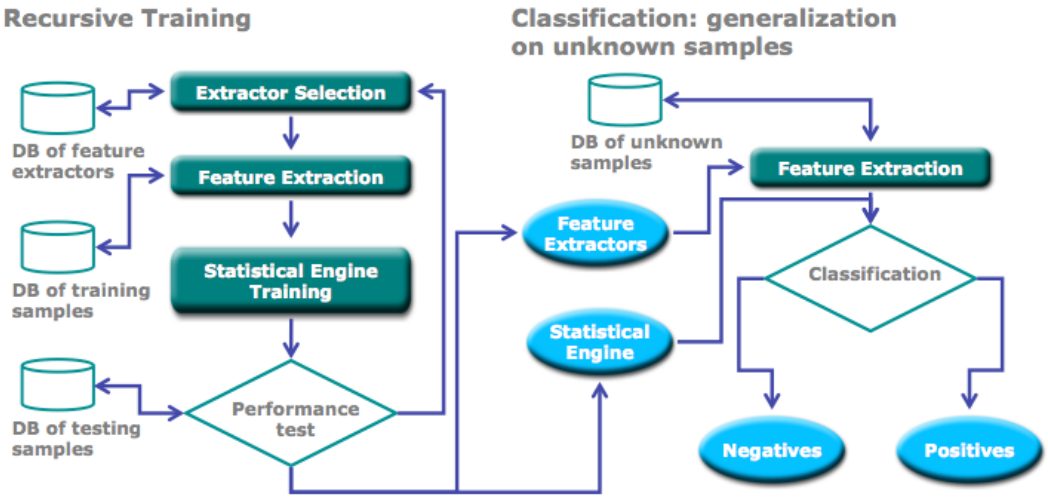


Figure 3-2: The classification module.

The classification process can be split in several phases:

- Extractor selection: it is the phase in which the features of interest are identified into the image.
- Extraction of the features over the entire image (a filter applied in a position of the image returns basically a floating point number that indicates the presence of a feature) from the set of possible feature extractor. For example a circular Lbp, 8 points, ray.
- Feature extraction: the feature extractor is applied to all positive (e.g. images with a face) and to all the negatives results (e.g. images without faces), obtaining for each image a vector of real numbers (the filter is applied in all possible positions inside the images).
- Statistical engine training: on the vectors of numbers belonging to the positive and negative classes the algorithm applies a statistical classification algorithm (it is based on support vector machines, neural networks, etc.).
- Performance test: this step checks whether the training produced generalizes specimens not seen during training. This control is performed applying it to positive and negative images (obviously not used in training). If the result is very bad then something went wrong during the training phase:
    - the training set was too small (there weren't examples of cases in the tests database),
    - the parameters of the static algorithm of the training are incorrect.
- Classification: the classification step is the generalization on unknown examples. If the result on the database of the test is good then you can use the results of the training phase (set of Extractors features and weights of the statistical algorithm training) on real cases:
    - the classification applies to the image of the real case the set of features extractors to obtain the vector of real numbers,
    - then, all the weights and / or statistical formulas of the training are applied to give a judgment of the image: positive or negative.

Coming back, the first module of the recognition procedure, the Face Detection module is based on two pipelined algorithms that are specialized for:

- Fast localization of image areas that are likely to contain faces.
    - Fast multi-scale and multi-location search.
- Accurate localization of facial features (eyes, mouth, eyebrows etc.).
    - Robust against unfavorable illumination conditions.
    - Robust against occlusions (dark sunglasses, scarves, etc.).

The following figure illustrates the output produced by the face detection module.

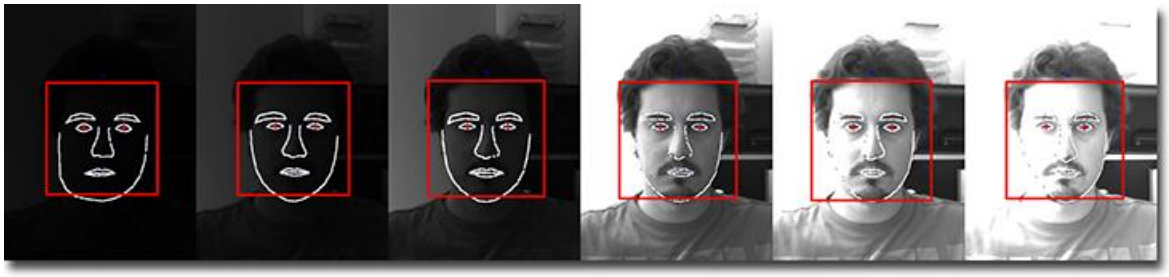


Figure 3-3: The output of the detection module.

During the data extraction and the training process of the face detection module, the algorithm uses a class template. The information in this template is organized by tag and is structured as follows:

- Position of Face Mask points.
- Photographic set: focus, illumination, background, etc.
- Subject: gender, age, ethnic group, etc.
- Face: pose, eye expression, gaze, mouth expression, etc.
- Morphology: eye type, lip type, nose type, mouth type, etc.

The following figure describes an example of the result of the matching and identification phase. The screenshot illustrates the face features identified in the photo and all the information extracted from the data base after the matching and identification process.

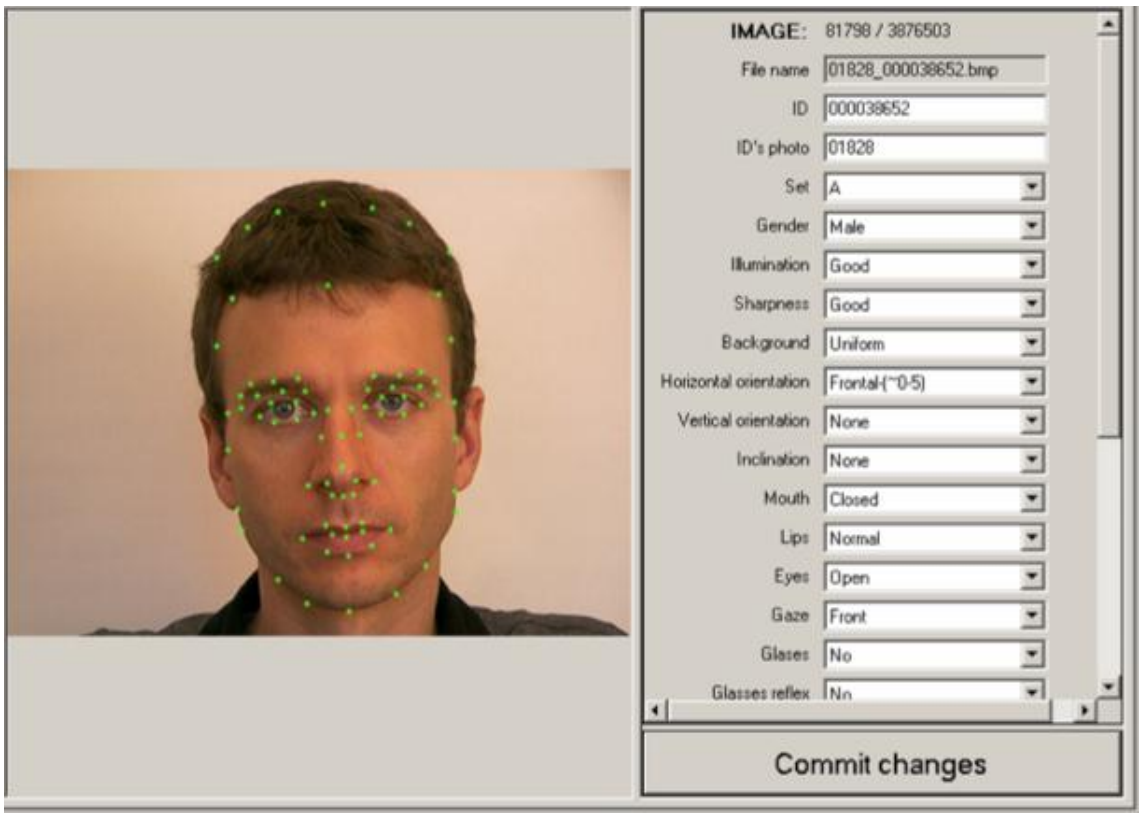


Figure 3-4: The matching and identification final result.

Finally, the database search is another important aspect of the recognition process. In this context, the law enforcement agencies use face recognition for automatic mug shot database reduction in their forensic investigative work.

The standard performance evaluation for face recognition system is the Face Recognition Grand Challenge framework, developed by the National Institute of Standards and Technology (NIST) in order to boost the research field of biometric face recognition.

The FRGC framework consists of:

- a face database - facial images with eyes position – built with a homogeneous criterion regarding sex, ethnicity and age, in order to obtain a significant sample of the actual worldwide population.
- A group of software tools, with a flexible XML configuration, to define, perform and evaluate standard test scenarios.

The following figure illustrates an example of the photo contained in a similar data base.

Figure 3-5: An example of photos in the database.

3.1.1.2 The Test Performed on the Recognition Software

The recognition software prototype has been tested with two different scenarios:

1. The first test considers optimal conditions both in terms of face position and aspect and in terms of illumination. These optimal conditions are ensured during the creation of the data base (enrol phase) and also during the operative phase, when people transit in front of the camera. In this test people have been obliged to have a collaborative approach that, practically speaking means that the person stands in front of the camera since the system hasn't detected that all the ICAO requirements are fulfilled. The photo taken in these conditions is the one that is used for the identification process.
2. The second test is characterized by more real conditions: the database is created using optimal condition, in order to guarantee a high quality master source, while the identification process takes place in a real environment, where the people are not aware of the presence of the recognition system.

This approach guarantees a good term of comparison between a real and an optimal environment and provides high value results because in the second test the people are not aware of the existence of the on-going experiment.

The following table illustrates the dimensions of the performed tests:

- target is the number of faces present in the database;

- query represents the number of people that have transited in front of the camera: during every transit the algorithm must compare the information extracted from the acquired photo with all the targets present into the database. This means that for 16028 transits, the algorithm performs 257 million tests.
- The last column reports the number of comparisons performed by the algorithm during the test.

Table 3-1: Dimensions of the Performed Tests.

Test	*Target (n. identities)*	*Query (n. identities)*	*N. of comparison tests (millions)*
1	*16028*	*16028*	*257*
2	*16028*	*8014*	*128*

The results of the two tests are illustrated in the following figure.

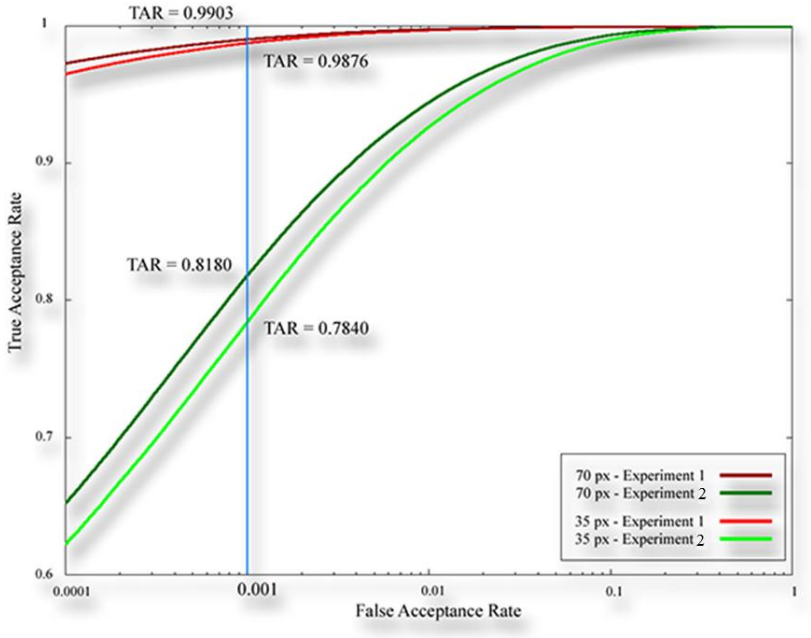


Figure 3-6: Test results.

In the previous figure, the TAR is the “true acceptance rate”, that is the rate of correct matching: a person present into the database passes in front of the camera and the system recognizes and identifies correctly the person. On the contrary, the FAR is the “false acceptance rate”, that is the rate of incorrect matching and identification: in this case a person that is not in the database passes in front of the camera and the system recognizes and identifies the person, making a mistake.

The blue vertical line represents the sensibility threshold of the recognition process. This threshold varies from 0 and 1, where:

- 0 means TAR = 1 and FAR = 1, that means that all the people passing in front of the camera are identified, both the ones present in database and the one not present (every matching value is greater than 1);
- “moving” the threshold to the left, raises the value of 1 and means that the FAR is lower (we want to avoid that people that are not “target” are wrongly recognized),

During experiment 1 the TAR value remains high since the value of 1 of the threshold, where the TAR is in any case 96%. On the contrary, in the experiment 2 the more the FAR is lowered (we don't want to recognize wrong people), the more the TAR is lowered and we risk to wrongly recognize people that are not present in the database.

The threshold that from the test better represents a good compromise between the two situations is the value of 0,001, where we have the probability that 1 every 1000 people is wrongly recognized and the 80% of the probability that the system identifies a person that is really present in the database. This situation is related to a single transit: if the first time a correct person is not recognized, the second time the same person has a high probability to be recognized. In other words, in front of a very rigid control on people not present in the database, we ask for some "patience" to people that are present in the database and are not identified at the first time. This is a very good result considering practical situation in a real environment.

Summarizing, the recognition software obtained the following good results:

- good performance with medium-to-low facial image resolution (70-35 px eye-to-eye distance),
- good performance with uncontrolled facial image,
- use of a small template size of 250 floating point array, 1.7 KB,
- fast template generation 25 biometric templates per second,
- fast similarity score generation 500.000 templates per second.

3.1.2 The Hardware Platform

The face recognition prototype is an all-in-one self-contained device capable to show and demonstrate, step by step, all the phases of the identification process of an individual. The prototype provides a touch screen and a simple interface that allows the user to interact with the recognition system during the entire recognition process, from the acquisition of the photo of a person to the final results of the identification. This interface has been developed only for demonstration purposes and will not be present in the final embedded system. The final prototype will be fully integrated in the camera enclosure and will provide a remote management interface.

The prototype is composed by the following main parts:

- main board,
- USB camera,
- white light illuminators,
- touch screen,
- custom case.

3.1.2.1 The Main Board

The prototype has been developed mainly to have a software development standard platform and in particular for demonstration purposes: it is intended to demonstrate mainly the features/functionalities of the recognition software and the quality of the results that it is capable to achieve. The presence of an extended user interface and the necessity of a development platform introduce new hardware requirements that oriented our choice to a standard embedded board that is more powerful than required. The prototype is based on our board called ANTARES i5 1GHz, shown below.

Figure 3-7: Eurotech ANTARES i5 1GHz.

The ANTARES main board is a 5.25" single board computer based on the new Mobile Intel QM57 Express Chipset. Designed to offer high performance with low power dissipation, the ANTARES is available with Intel Core i7 or Core i5 processor options and is ideal for use in compact spaces with restricted ventilation.

Traditional embedded system requirements are well catered for with features such as Watchdog, GPIO, four serial ports, and an integrated SD/MMC Flash port for accessing storage cards and Mini PCIe expansion for wireless modules such as Wi-Fi, Bluetooth and cellular modems. Features such as virtualisation, hyper threading, remote system management via Intel's AMT feature (available even if the operating system has crashed), provide solutions for many of the key issues being considered in newer embedded designs. Coupled to a wide operating temperature range, options for fanless operation, and long term chipset support, the ANTARES embedded PC is ideal for applications in commercial, industrial and transportation projects requiring high performance computing within a rugged and reliable platform.

This platform is perfectly suited for development purposes and can support the extended requirements of a demonstrator.

Although a board with a Intel Core i5 could not be considered an embedded system the software and the algorithms are designed to be optimized for the future to run in an embedded system: this is the real objective of our effort. The goal is to port the application on a custom board based on the system on chip manufactured by Texas Instruments and based on ARM architecture: the DM816x DaVinci system on a chip. The board has been conceived specifically for video processing and provides natively hardware encoders and a digital signal processor (see figure below).

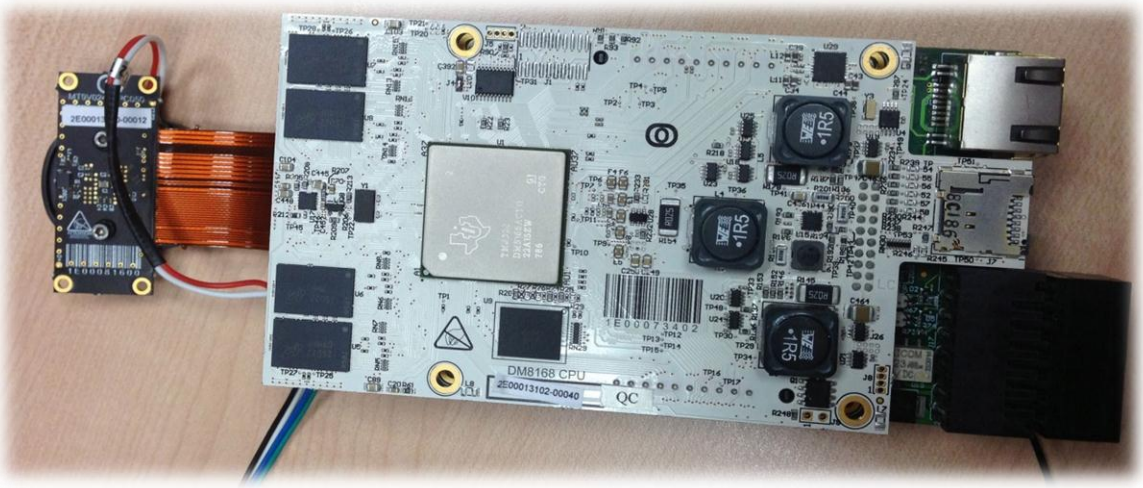
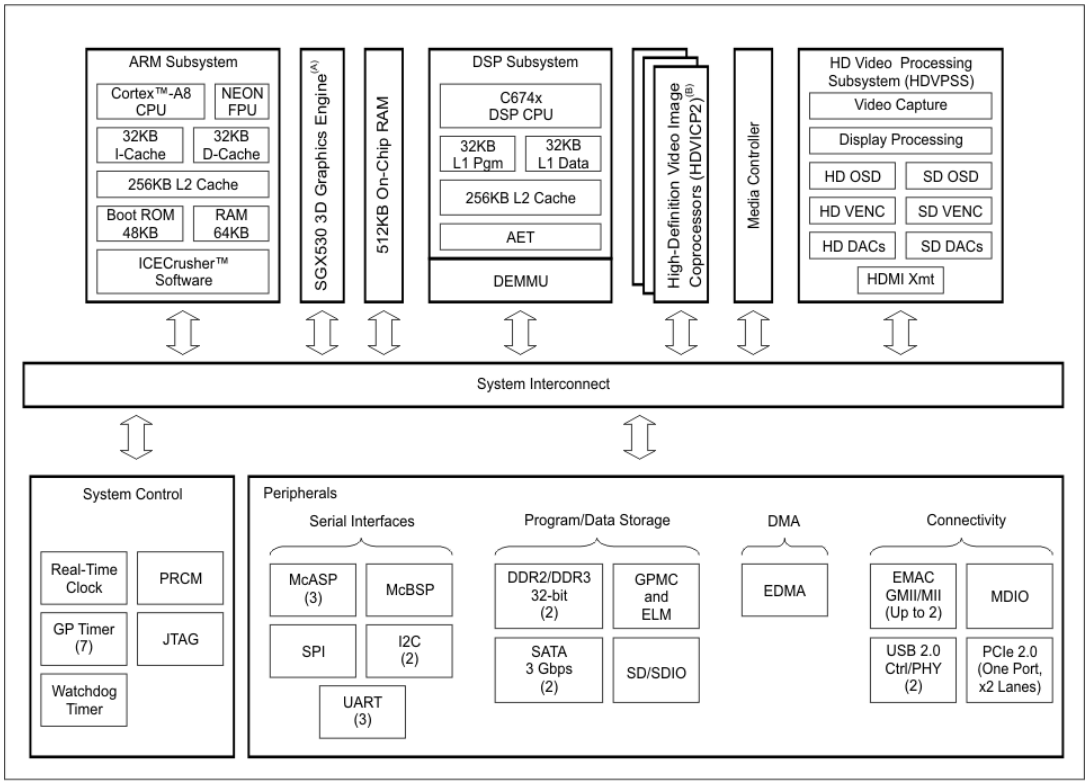


Figure 3-8: The custom embedded board based on TI SoC.

The key elements of the DM816x DaVinci are the three high-definition video and imaging coprocessors (HDVICP2). Each coprocessor can perform a single 1080p60 H.264 encode or decode or multiple lower resolution or frame rate encodes and decodes. It provides also TI C674x VLIW floating-point DSP core, and high-definition video and imaging coprocessors and multichannel HD-to-HD or HD-to-SD transcoding along with multi-coding are also possible. This solution has been conceived specifically for demanding HD video applications. The main architecture of the system on chip (SoC) is described in the figure below:



A. SGX530 is available only on the TMS320DM8168 and TMS320DM8166 devices.
B. Three HD Video Image Coprocessors (HDVICP2) are available on the TMS320DM8168 and TMS320DM8167 devices; two are available on the TMS320DM8166 and TMS320DM8165 devices.

Figure 3-9: The embedded board architecture.

The block diagram of the components of the system on chip is described in the figure below. The figure illustrates also how the various units of the SoC cooperate together to provide high level imaging and video performance in a real embedded system board.

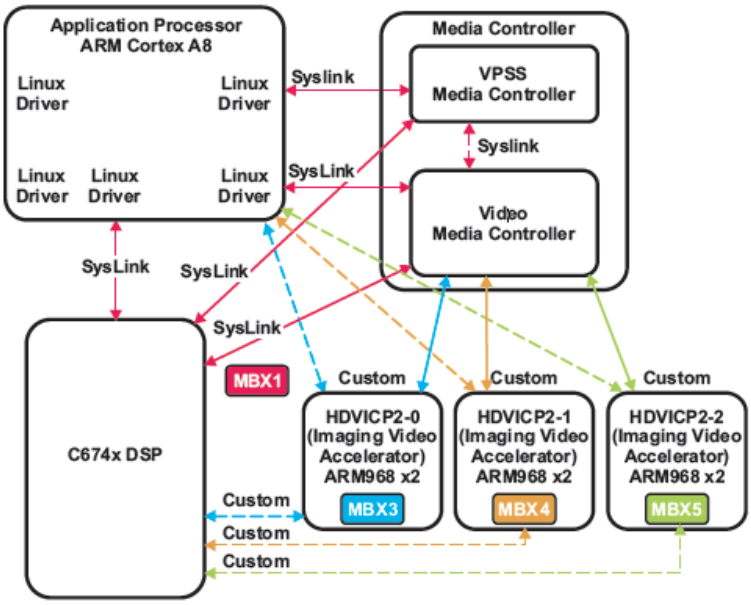


Figure 3-10: The SoC architecture.

The technical specifications of the DM816x are:

- ARM® Cortex-A8 RISC Processor
  - Up to 1.35 GHz
- C674x VLIW DSP
  - Up to 1.125 GHz
  - Up to 9000 MIPS and 6750 MFLOP
- ARM Cortex™-A8 Core
- ARMv7 Architecture
  - In-Order, Dual-Issue, Superscalar Processor Core
  - NEON Multimedia Architecture
  - Supports Integer and Floating Point (VFPv3-IEEE754 compliant)
  - Jazelle RCT Execution Environment
- ARM® Cortex™-A8 Memory Architecture
  - 32K-Byte Instruction and Data Caches
  - 256K-Byte L2 Cache
  - 64K-Byte RAM, 48K-Byte Boot ROM
- TMS320C674x Floating-Point VLIW DSP
  - 64 General-Purpose Registers (32-Bit)
  - Six ALU (32-Bit and 40-Bit) Functional Units
  - Supports 32-Bit Integer, SP (IEEE Single Precision, 32-Bit) and DP (IEEE Double Precision, 64-Bit) Floating Point

  - Supports 32-Bit Integer, SP (IEEE Single Precision, 32-Bit) and DP (IEEE Double Precision, 64-Bit) Floating Point
  - Supports up to Four SP Adds Per Clock and Four DP Adds Every Two Clocks
  - Supports up to Two Floating-Point (SP or DP) Approximate Reciprocal or Square Root Operations Per Cycle
- Up to Three Programmable High-Definition Video Image Coprocessing (HDVICP2) Engines
  - Encode, Decode, Transcode Operations
  - H.264, MPEG2, VC1, MPEG4 SP and ASP
- SGX530 3D Graphics Engine (available only on the DM8168 and DM8166 device)
  - Delivers up to 30 MTriangles per second
  - Universal Scalable Shader Engine
  - Direct3D Mobile, OpenGL ES 1.1 and 2.0, OpenVG 1.1, OpenMax API Support
  - Advanced Geometry DMA Driven Operation
  - Programmable HQ Image Anti-Aliasing
- HD Video Processing Subsystem (HDVPSS)
  - Two 165 MHz HD Video Capture Channels
  - One 16-Bit or 24-Bit and One 16-Bit Channel
  - Each Channel Splittable Into Dual 8-Bit Capture Channels
  - Two 165 MHz HD Video Display Channels
  - One 16-Bit, 24-Bit, 30-Bit Channel and One 16-bit Channel
  - Simultaneous SD and HD Analog Output
  - Digital HDMI 1.3 transmitter with PHY with HDCP up to 165-MHz pixel clock
  - Three Graphics Layers
- Dual 32-Bit DDR2 and DDR3 SDRAM Interfaces
- Supports up to DDR2-800 and DDR3-1600
- 2 GB Total Address Space
- One PCI Express (PCIe®) 2.0 Port With Integrated PHY
- Serial ATA (SATA) 3.0 Gbps Controller With Integrated PHYs
- Two 10 Mbps, 100 Mbps, and 1000 Mbps Ethernet MACs (EMAC)
- IEEE 802.3 Compliant (3.3V IO Only)
- Dual USB 2.0 Ports With Integrated PHYs
  - USB 2.0 High-Speed and Full-Speed Client
  - USB 2.0 High-Speed, Full-Speed, and Low-Speed Host
- General Purpose Memory Controller (GPMC)
- Flexible Asynchronous Protocol Control for Interface to FPGA, CPLD, ASICs
- Enhanced Direct-Memory-Access (EDMA) Controller
- Seven 32-Bit General-Purpose Timers
- One System Watchdog Timer

- Three Configurable UART, IrDA, and CIR Modules
- Three Multichannel Audio Serial Ports
- Multichannel Buffered Serial Port (McBSP)
- Real-Time Clock (RTC)
- Up to 64 General-Purpose IO (GPIO) Pins
- On-Chip ARM® ROM Bootloader (RBL)
- Power, Reset, and Clock Management
- 1031-Pin Pb-Free BGA Package (CYG Suffix), 0.65-mm Ball Pitch
- 40-nm CMOS Technology
- 3.3-V Single-Ended LVCMOS IOs (except for DDR3 at 1.5 V, DDR2 at 1.8 V, and DEV_CLKIN at 1.8 V)

3.1.2.2 The USB Camera

The camera is an important element for the recognition system. The prototype has been equipped with an USB camera with autofocus and HD resolution. The selected model is the Microsoft Lifecam Studio HD shown below.

Figure 3-11: The USB camera.

This camera provides a simple solution for the prototype and the use of the USB connector represents an easy way to try different type of cameras with different characteristics, an aspect very important from a development point of view. Furthermore, the use of a commercial standard camera demonstrates the qualities of the algorithm of face recognition that, can obviously have a benefit from the use of a custom camera, but doesn't necessarily require custom hardware to satisfy the quality level required by the standards. It is important to point out that this important result has been obtained with an embedded system.

The technical specifications of the USB camera are:

- Sensor HD 1080 pixel, with exceptional sharpness and image quality.
- Glass lens for the highest precision and ensures sharpness images.
- TrueColor technology. Automatically controls the exposure to ensure bright colors and vivid video.
- USB 2.0 data connection.

The embedded board that will be used in the final implementation integrates the camera that is based on a high quality CMOS CCD sensor. This approach allows to directly accessing the raw data acquired by the sensor, avoiding the compression and decompression of the images acquired by the camera connected via USB. Furthermore, the board will allow the possibility to use different type of sensors, depending on the requirements of the application scenario. It will be possible to select the sensor depending on the resolution, on the pixel size, on the frame rate, etc.

The development of the board will support initially two kinds of sensors:

- the Aptina AR0331: an image sensor that integrates a small, high performance global shutter technology for high speed image capture into a 1/3-inch optical format high definition (HD) device. It provides a 3.75-micron global shutter pixel with low light performance that can stop action without the artifacts typically associated with conventional rolling shutter pixels.
- The Aptina MT9P001I12STC: an HD CMOS Sensor with a pixel size of 2.2 um, low-light sensitivity and low noise level. This sensor enables high speed image capture capabilities and includes variable functions, such as gain, frame rate, and exposure while maintaining low power consumption.
- Optionally, if the application context requires high quality images, the board will be able to support also the the Aptina MT9M031: this is a full HD, 3.1-megapixel sensor based on new 2.2-micron pixel. The AR0331 targets the mainstream 1/3-inch optical format surveillance camera market with excellent image quality. It provides full HD video with wide dynamic range (WDR) capability and built-in adaptive local tone mapping. The new sensor is capable to work up to 1080p at 60 fps, adopting a technique to enable the sensor's sub 1-lux low light performance.

The technical specifications of the sensors are provided in the following.

3.1.2.2.1 AR0331 Technical Specifications:

- Array
  - Res. 1.2MP
  - Optical Format 1/3"
  - Active Array 1280x960
  - Imaging Area 1280x960
- Sensitivity
  - Pixel Size 3.75
  - Dynamic Range 83.5dB
  - Responsivity 8.5V/lux-sec (Monochrome)
- Speed/Output
  - Frame Rate 60 fps at 720P HD mode
  - Chroma RGB and Mono Chrome
  - Shutter type Global Shutter
  - Data Rate 60 frame per second at 720P
  - Master Clock 74.25Mhz
  - Data Format Progressive Scan
- Environmental
  - Supply Voltage 1.8V/2.8V
  - Power Consumption 270mW

- Operating Temperature -30°C to + 70° C
- ROHS Compliance Yes

- Package
  - Package iLCC
  - Package Dimension 10x10mm

3.1.2.2.2 MT9M031 Technical Specifications:

- Array
  - Res. 3.1MP
  - Optical Format 1/3-inch
  - Active Array 2052x1536
  - Imaging Area 2048x1536
- Sensitivity
  - Pixel Size 2.2µm
  - Dynamic Range Up to 100 dB
  - Responsivity 1.9 V/lux-sec
- Speed/Output
  - Frame Rate 60 fps
  - Chroma RGB
  - Shutter type ERS
  - Master Clock 74.25Mhz
- Environmental
  - Supply Voltage 1.8V/2.8V
  - Power Consumption <720mW at full frame rate/speed
  - Operating Temperature -30C + 85C
  - ROHS Compliance Yes
- Package
  - Package iLCC and iBGA
- Package Dimension 10x10mm and 9x9mm

3.1.2.2.3 MT9M031 Technical Specifications:

- Array
  - Res. 5MP
  - Optical Format 1/2.5 inch
- Sensitivity
  - Pixel Size 2.2µm
- Speed/Output
  - Frame Rate 15 fps

- Frame Rate Video 720p30 HD
- Chroma RGB

- Package
  - Package iLCC

3.1.2.3 The White Light Illuminator

The conditions of light represent an important factor in face recognition, significantly influencing the results and the quality of the identification process.

The vertical light coming from the ceiling lights creates, for its own nature, shadows on the face, especially under the eyes and the noise. These shadows disturb the correct face analysis and for this reason should be eliminated during the recognition process. To solve this issue the prototype is equipped with two illuminators that illuminate the front face and completely eliminate shadows.

The two illuminators have been developed using six Osram leds for each illuminator. The Osram leds have the following technical specifications:

- Model: Osram LUW W5PM.
- Illuminance: the simulation of the illuminance made by the Zemax Optical Cad is described in the following figures.

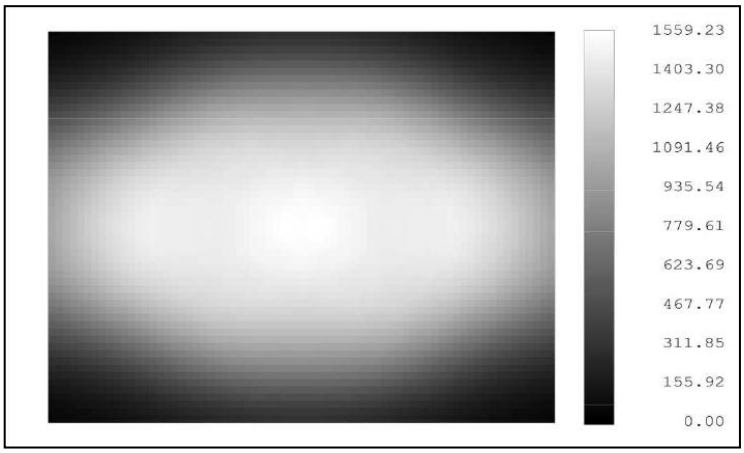


Figure 3-12: Luminance at 1 m of distance.

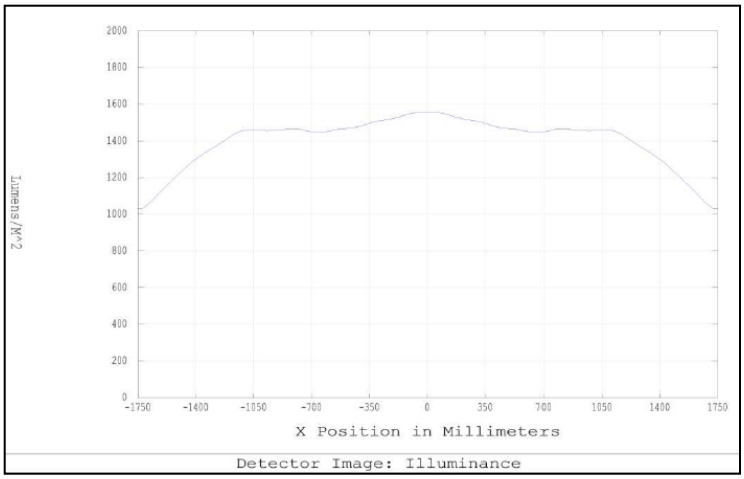


Figure 3-13: X profile.

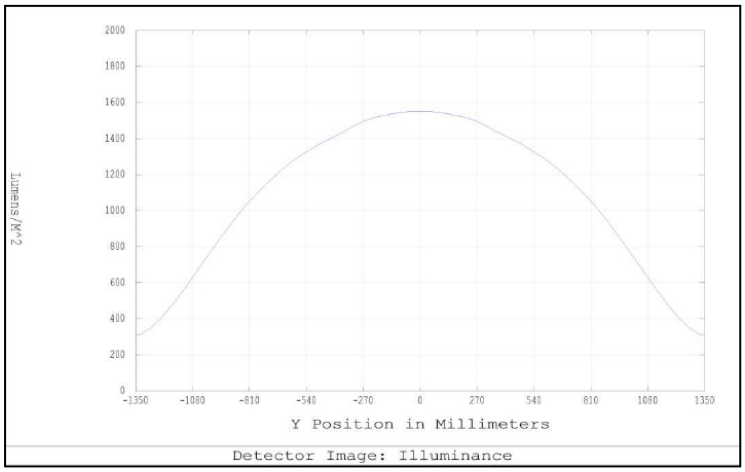


Figure 3-14: Y profile.

3.1.2.4 The Custom Enclosure

Considering the demonstrator objectives, the prototype has been developed adopting a custom enclosure that offers the possibility to show the functionalities of the identification system, maintaining small dimensions and portability. The enclosure has been manufactured with aluminium and is described in the following figures.

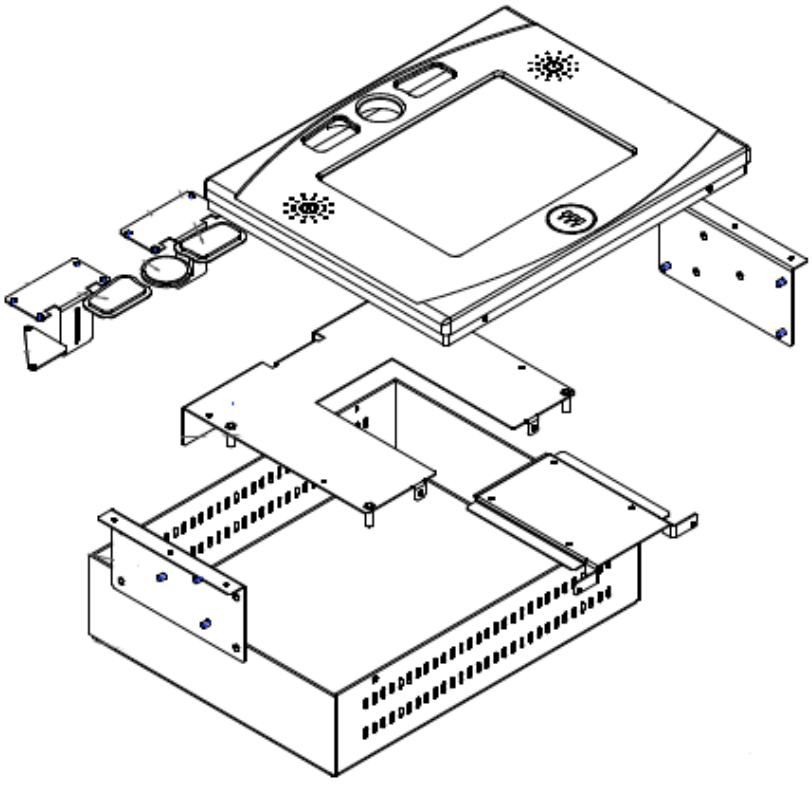


Figure 3-15: The parts of the enclosure prototype.

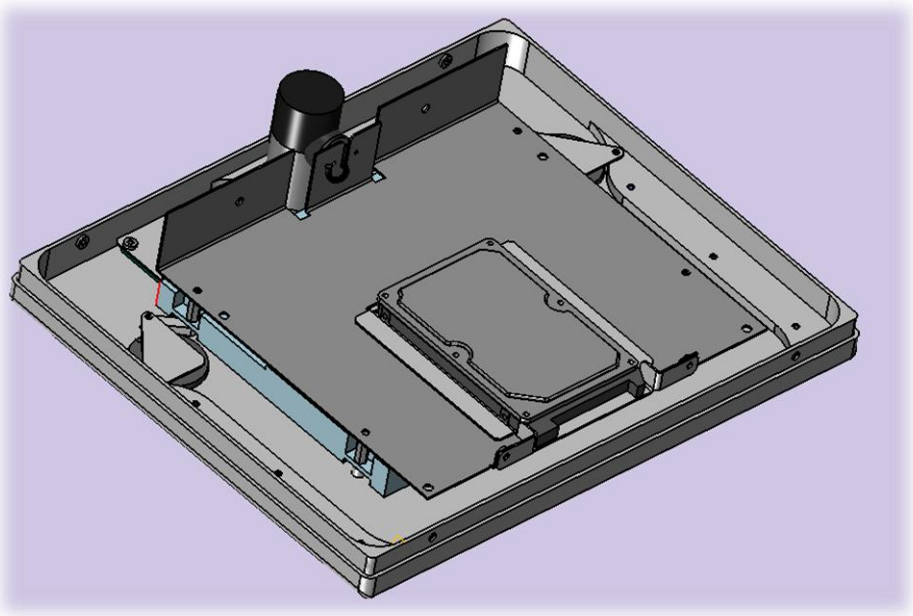


Figure 3-16: Rendering of the bottom cover of the enclosure.

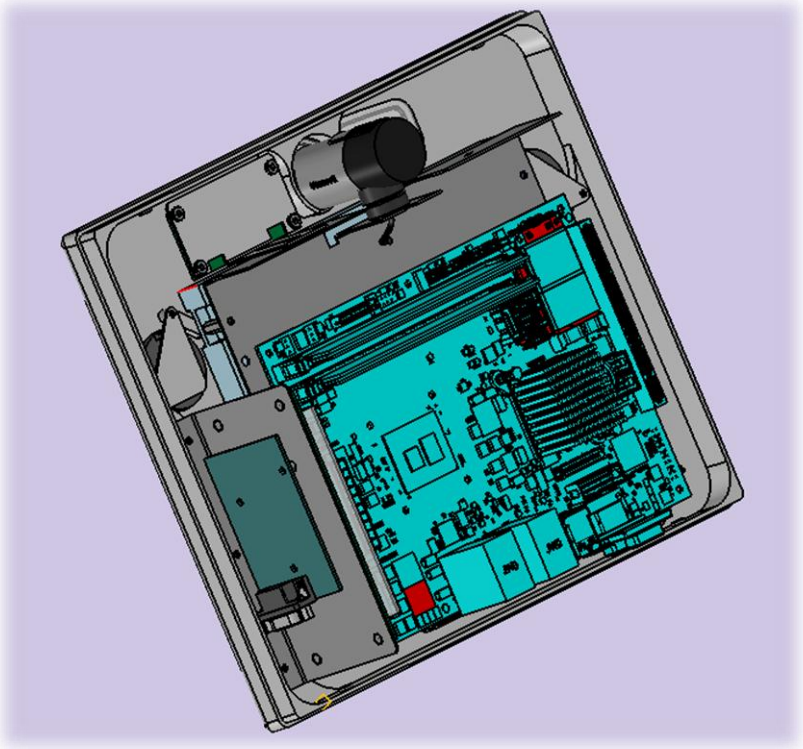


Figure 3-17: Rendering of the top cover of the enclosure.

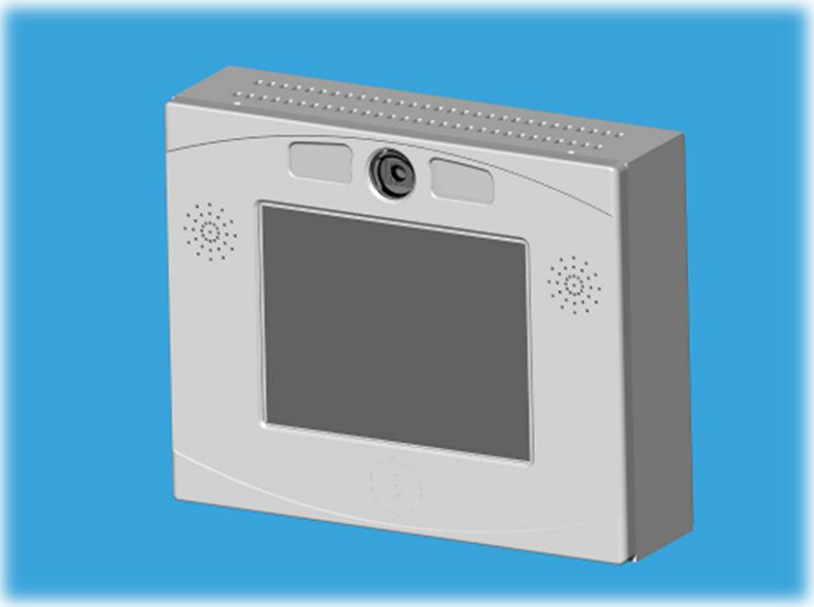


Figure 3-18: Rendering of the enclosure.

Figure 3-19: The real prototype.

4 Power Node

The power node technology prototype available is a mechanism that utilizes the GPU to accelerate hashing and hash lookup.

4.1 GPU Accelerated Hashing and Hash Lookup Mechanism

This implementation involves the development of a lightweight, efficient, GPU accelerated hashing and hash lookup mechanism utilizing the CUDA GPGPU toolkit.

The system will maintain a hash-table with the hash values of known files. Upon execution, it will compute the digest of input files using the same hash function and, depending on application, look for a match or a mismatch between pre-existing and calculated values.

The system will exploit the CUDA toolkit and parallelization capabilities of modern GPUs to improve the performance of three main functionalities:

- Digest computation.
    - The inner calculations of the chosen cryptographic hash function will be run on the GPU.
- Equality check of two digests.
    - Parallel threads will be used to examine different segments of the two digests.
- Lookup operation of the hash table with the file digests.
    - The GPU will be used to optimize this operation. Different threads will simultaneously examine different buckets of the hash table. An important factor is the selection of a proper hash function for the hash table construction.

Possible applications of the mechanism include (but are not limited to) malware detection on local disks, network traffic monitoring and file integrity checks for pre- or post-installation audits.

This scheme and its variations will further enhance the functionality and performance of next generation nSHIELD power nodes, taking advantage of state-of-the art GPU-accelerated ARM-based systems (e.g. NVIDIA Carma platform [12]).

5 Dependable self-x Technologies

The dependable self- technology prototypes available are the following:

- The hardware platform to be used as a basis for development.
- An anonymity service targeting applications with demanding privacy needs.
- Mechanisms for DoS/DDoS attack mitigation.

5.1 Hardware platform

According to the task requirements the research was focused to define a prototype of a scalable node family in order to cover the three node typologies and to be useful for the four application scenario.

The selected platform has the following main features:

- Multi-core platform including micro controller, FPGA and DSP.
  - Scalable for the three node typology.
  - Low dimension for embedded system.
  - Reconfigurable and reprogrammable in order to meet metrics requirements.
- Micro controller can be used with different Operating Systems (Linux and Windows CE)
  - Possibility to use commercial and free tools chain.
  - Possibility to work in different scenario.
- Hardware re-configurability (FPGA) allows to change the node characteristic according to the application (i.e.: surveillance, voice/facial recognition, avionic, social mobility and networking) and to self-recovering and re-configurability issues.
- DSP allows improving the node computational performances in order to speed up operations in mobile systems with an high level of re-configurability.
- The several serial ports allow to implement self-adaptation of sensing tasks and to use the platform for the project scenarios.
- OMBRA tool chain:
  - µProcessor with two boot mode available allows to use either the windows CE tool or the linux gnu based tool set.
  - FPGA can be managed by Xilinx tool.
  - OMBRA support: it is a platform owned by an nSHIELD partner interested to the deployment and to support users.

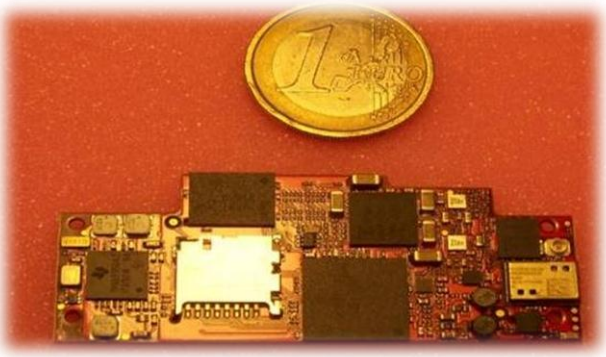


Figure 5-1: The OMBRA board.

5.1.1 Demonstration

This chapter describes the implementation on the OMBRA platform of the Montgomery's algorithm that performs the point multiplication on an elliptic curve.

The target is to show the running algorithm on the implemented embedded Linux platform (Linux omap 2.6.32) and calculate the working time. It was needful to compile for the above mentioned Linux version the gmp libraries.

In order to achieve a high accuracy in the running time calculation the algorithm is repeated 1000 times, dividing the final result for the repetition number. In this way we can obtain a reasonable average running time for the implemented algorithm.

We implemented the demo for two NIST standard elliptic curve: secp256r1 and secp521r1.The first curve is 256 bit length instead the second one is 521. The test program first of all asks to select the curve, and then the two branches run independently.

In the 256 bit length case the time to perform the point multiplication (change coordinates included) is 0.023390 seconds. In the 521 bit length case the time to perform the point multiplication (change coordinates included) is 0.113280 seconds.

We perform the same experiments on an intel core i7 @3Ghz to compare the results. In the 256 bit length case the time to perform the point multiplication (change coordinates included) is 0.003895 seconds. In the 521 bit length case the time to perform the point multiplication (change coordinates included) is 0.017379 seconds. We can note that the computational times on the OMBRA platform are very good (ratio is around 6 in the 256 bit length and around 6,5 in the 521 bit length) considering that its processor runs at 1Ghz.

The figure below shows the algorithm execution.

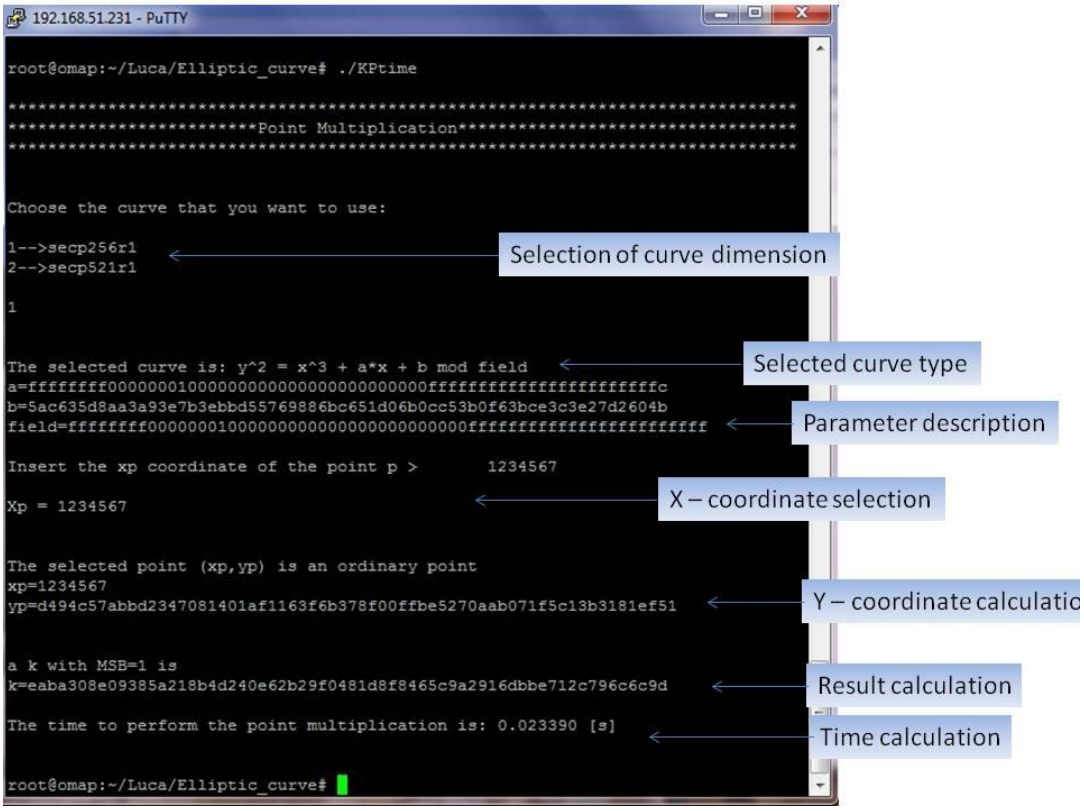


Figure 5-2: Algorithm execution.

5.2 Anonymity and location privacy service

An embedded system might be associated with a user, e.g. a mobile phone, or a system, e.g. a wireless sensor node, the location of which might necessary to access certain location-based services (LBSs) On the other hand, explicitly identifying its location might reveal information about the user's location. Although this might not constitute a security risk for a system, it raises many privacy concerns about disclosure of sensitive information (e.g. daily schedule, medical conditions and political affiliations) to unauthorised parties.

Location privacy can be typically protected through anonymity and cloaking services (see the figure below), i.e. hiding a node's location among a set of others and in a larger region respectively, although in many cases simply hiding the node's identity does not safeguard its location privacy [13]. Certain nSHIELD deployment scenarios would welcome such a feature (e.g. wearable/personal nodes) and, in such cases, an anonymizer will have to be deployed that will either decentralise data, encrypt them, change the traffic pattern, flood the network or implement a "K-anonymity" protocol for location cloaking. The latter privacy concept requires that an entity's location information is sent in such a form that makes said entity indistinguishable from K-1 other neighbouring entities. Naturally, a lower K value provides less privacy protection but also less communication overhead and better quality of monitoring and the opposite stands for higher K values. The level of anonymity (K-level) required should be modifiable, depending on the context (thus dictated by the active security policy), and any such changes should be communicated to the nodes by the location-information collection point (i.e. server).

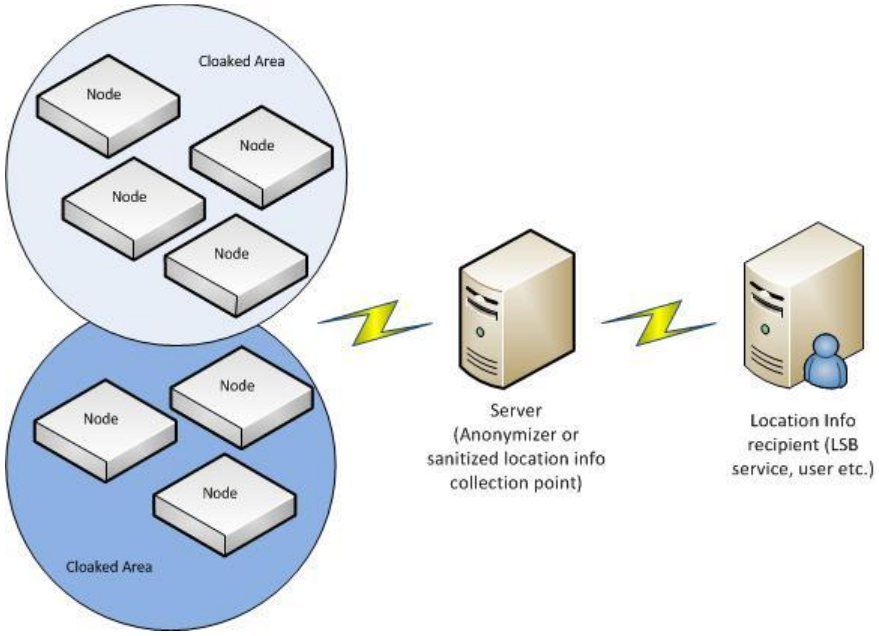


Figure 5-3: Generic anonymization service architecture.

An anonymizer component will be developed part of the nSHIELD network layer although the computation overhead incurred by these services might prohibit its deployment in any type of ESD.

After an extensive state of the art review, the TinyCasper [14] scheme was chosen and will be implemented, aiming to preserve personal location privacy via the well-established K-anonymity privacy concept, while enabling the system to provide location monitoring services.

To facilitate the deployment on resource constrained devices and heterogeneous environments, it was decided to avoid assumptions about the capabilities of devices that will have to run the anonymity service. Thus, the resource-aware version of the original scheme was implemented. Choosing a lightweight scheme produces a universal solution which can be run in heterogeneous systems, from highly resource constrained devices like wearable units, wireless sensors and various embedded systems to high-power

devices like modern smart phones. There are no prerequisites regarding network topology since communication is based on a distributed tree; only a path from each node to the server is needed.

The scheme developed is not limited to location monitoring of objects (as in the original TinyCasper). Nodes can also act as proxies to serve requests of the objects (i.e. users) to services requiring their location. The introduction of this reverse information flow, i.e. having users forward their location-related requests to a node in range, allows the utilization of relevant services by the users without disclosing the true identity and exact location. In fact, the nodes could be actual users of the service themselves; users who opt to allow their devices to act as proxies for other users of the anonymity service. This concept is similar to how the TOR network operates, where the users can, upon installation, choose if they want to join the anonymization network as simple clients or as clients and proxies, helping anonymize the traffic of other TOR users.

This significant modification on the original system necessitated the introduction of certain refinements in parts of the service. The latter was also extended to cater for nodes with different capabilities as various users may have heterogeneous devices. More importantly, provisions have been made for mobile nodes (e.g. wearable nodes), moving away from the static infrastructure deployment of the original service. To enhance this functionality, special features present on the test platforms were exploited, namely using an accelerometer to detect when the node is mobile and trigger the relevant events. Finally, a corresponding application was developed on the server side, featuring a graphical user interface (GUI) to facilitate system monitoring.

5.3 Automatic Access Control

Access control mechanisms are in charge of preventing malicious entities to access the physical resources of a network node. Nodes utilize asymmetric cryptography to verify access requests. A DoS attack can be performed if a large number of access requests are sent to exhaust node's resources. Automatic access control embodies some lightweight features to easy the verification process and avoid the DoS attack. Such features include hash functions, matrix multiplication, pseudorandom number generators and cyclic redundancy check (CRC). For example, a client sends a hashed secret that is already known to the server. The server keeps a map for the hashed secrets for all its clients. Then the server simply looks up for the secret to quickly verify if the node is legitimate and counter the DoS attack efficiently. The failure of some access control requests could be an indication that the system is under attack. The relevant information can be reported to an IDS.

Several automatic access control protocols have been proposed. They try to provide security properties like mutual authentication, forward security and anonymity. Also they try to counter DoS, replay and de-synchronization attacks to the protocol steps as well as link-ability of different communications of the same user.

Gossamer is a protocol for preventing DoS attacks on RFID systems. It belongs to the UMAP (Ultra Lightweight Mutual Authentication Protocol) family of protocols. A reader uses index-pseudonyms to retrieve tag information. Readers and tags share sub keys which are part of a single key. Then these sub keys are used to build the messages exchanged in the mutual authentication process. The protocol is based on bitwise logical operations such as XOR, OR, and AND. The reader generates a pseudorandom numbers and tags use them to create messages. Gossamer is vulnerable to a DoS attack by de-synchronization.

For nSHIELD dependable self-x technologies, we propose a variant of the Gossamer protocol. The new protocol will be secure against the attacks of the pure Gossamer. We will apply the protocol in BeagleBones/BeagleBoards. A BeagleBoard acting as a server and a small number of BeagleBones acting as tags will connect to the board. The protocol will be used for node and network protection against DoS attacks.

5.4 DDoS Attack Mitigation on SPD Power/Micro Nodes

Distributed Denial of Service attacks are one of the most common weapons in cyber-crime. We now face the reality of botnets that hold millions of compromised systems ready to cripple almost any network or

internet service they choose. Due to the specification of the IP protocol, those systems can generate any kind of packets. Those packets can contain any random source IP address. Packets that originate from compromised systems as part of a DDoS attack usually hold a random source IP address or an address that belongs to another host. This technique, called IP spoofing, is one of the reasons that make DDoS attacks very difficult to be traced and/or countered.

nSHIELD's power and, to an extent, micro nodes have the required characteristics to be considered as service providers. This brings those systems to the same environment with conventional service providers and makes them potential targets of DDoS attacks.

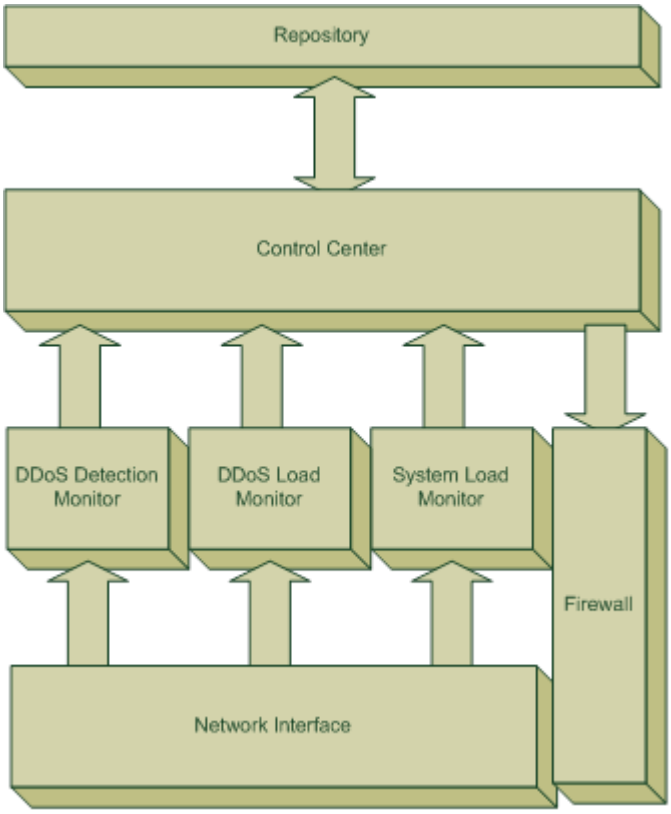


Figure 5-4: The filtering and traceback mechanism architecture.

We designed a packet marking scheme in conjunction with an intelligent filtering and traceback mechanism that can effectively stop ongoing DDoS attacks. It is a highly configurable mechanism able to react in different attack scenarios and ensure the highest amount of legitimate user service under an ongoing DDoS attack. With precise tuning of this mechanism, an organization can provide robust while flexible protection against such attacks.

The mechanism consists a DDoS load monitor, a DDoS detection monitor, which can be part of existing network intrusion detection (NIDS) system, a system load monitor, a reconfigurable firewall able to handle packet markings, a repository and a control centre.

In general, the two DDoS monitors observe incoming traffic and send alerts to the control centre. The control centre gets periodic reports from all three monitors. The repository holds information about the networks that are of interest to the victim's company or organization. In the case of an ongoing DDoS attack, the control centre evaluates the reports and, in conjunction with the information in the repository, sends directives to the firewall regarding which networks (i.e. markings) it should filter out.

6 Cryptographic technologies

The cryptographic technology prototypes available are the following:

- A cryptographic library that represents the implementation of the point multiplication operation on an elliptic curve on prime field, which is the core of elliptic curve-based cryptographic protocols like Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA). The elliptic curve point multiplication operation is defined as Q=kP, where Q, P are points in a previously chosen elliptic curve defined over a prime field GF(p), and k is a field element. Here p is the characteristic of the field.
- A compact crypto library for a subset of lightweight ciphers and compact implementations of standard ciphers. The library utilizes open source implementations of known ciphers. In this first prototype, the library contains block/stream ciphers and hash functions. For block ciphers, it supports AES, DES, 3DES, PRESENT, LED, KATAN, KTANTAN, Clefia, Camellia, XTEA and XXTEA in ECB, CBC and CTR modes of operation with the padding schemes zeroPadding, PKCS5, PKCS7, ISO_10126-2, ISO_7816-4 and X9.23. For stream ciphers, it supports the ARC4 and the eSTREAM project finalists Salsa20, Rabbit, HC128, SOSEMANUK, Grain, Grain-128, Trivium and MICKEY v2. For hash functions, it supports the MD5, SHA-1, SHA-2, the new SHA-3 function Keccak and the other finalists of the SHA-3 contest Blake, JH, Groestl and Skein.
- Key exchange protocols.

6.1 Elliptic Curve Point Multiplication over Prime Fields Library

The elliptic curves supported are in the simplified Weierstrass form $y^2=x^3+ax+b$, where a,b are elements of GF(p) with p different from 2 or 3. Supported curves include those specified by NIST (Standards for Efficient Cryptography 2 – September 20, 2000, available at www.secg.org) indicated as secp160r2, secp192k1, secp192r1, secp224k1, secp224r1, secp256k1, secp256r1, secp384r1, secp521r1.

This library supports elliptic curve point representations called Affine coordinates and Jacobian coordinates.

This library employs the Montgomery Ladder algorithm in order to accomplish the elliptic curve point multiplication. This is the most efficient algorithm producing side-channel attacks-resistant cryptosystems. For testing purposes only, the elliptic curve point multiplication is provided with the trivial binary method also.

This library requires a Linux operating system and the GNU Multiple Precision Library (libgmp) version 4.3 or higher.

The following flow chart represents the structure of the library:

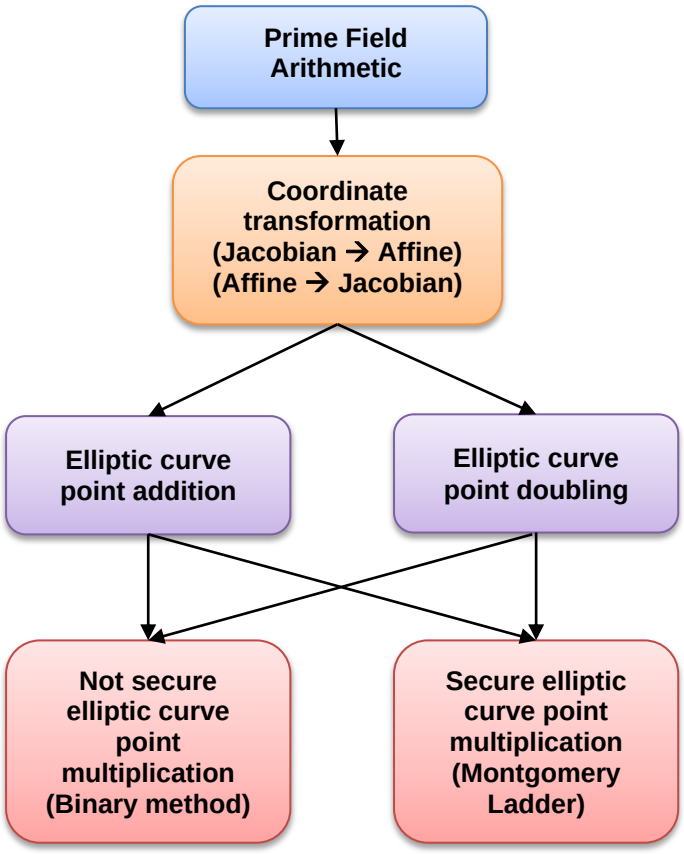


Figure 6-1: Library structure.

This library can be used to implement the elliptic curve diffie-hellman algorithm or the elliptic curve digital signature algorithm: it is a horizontal SPD technology; in fact it can be used in many applications and scenarios. For example it can be used in scenarios that involving secure communications and secure data exchange on embedded systems. In particular it is tested on the OMBRA platform with a Linux operating system. This platform presents a SOIC integrating an ARM Cortex-A8 running at 1GHz.

6.2 Compact Crypto Library

Well-known crypto libraries, like openSSL, target on mainstream applications. Such libraries support high levels of security and don't take into consideration the special needs of constrained and ultra-constrained devices. Other libraries that are designed for embedded system applications support crypto-primitives that are designed for embedded devices and are optimized for space, speed or power consumption. On the other hand, such libraries either contain redundant functionality as they support a wide range of cryptographic primitives or contain a small number of primitives.

For nSHIELD cryptographic technologies, we implement a compact crypto library for a subset of lightweight ciphers and compact implementations of standard ciphers. The library utilizes open source implementations of known ciphers that are applicable to constrained devices. We implement a common API for utilizing all of them with their different parameters. Our novelty is the segmented compilation. A user can define an exact set of crypto-primitives that will be compiled without requiring compiling the whole library. Thus, the executable file that will be running on the embedded devices will be small. For example, a user can choose to compile only a compact AES implementation and use it through the common API for block ciphers. As embedded devices will run a specific set of protocols and crypto-primitives, our segmented implementation of lightweight primitives is a good candidate library.

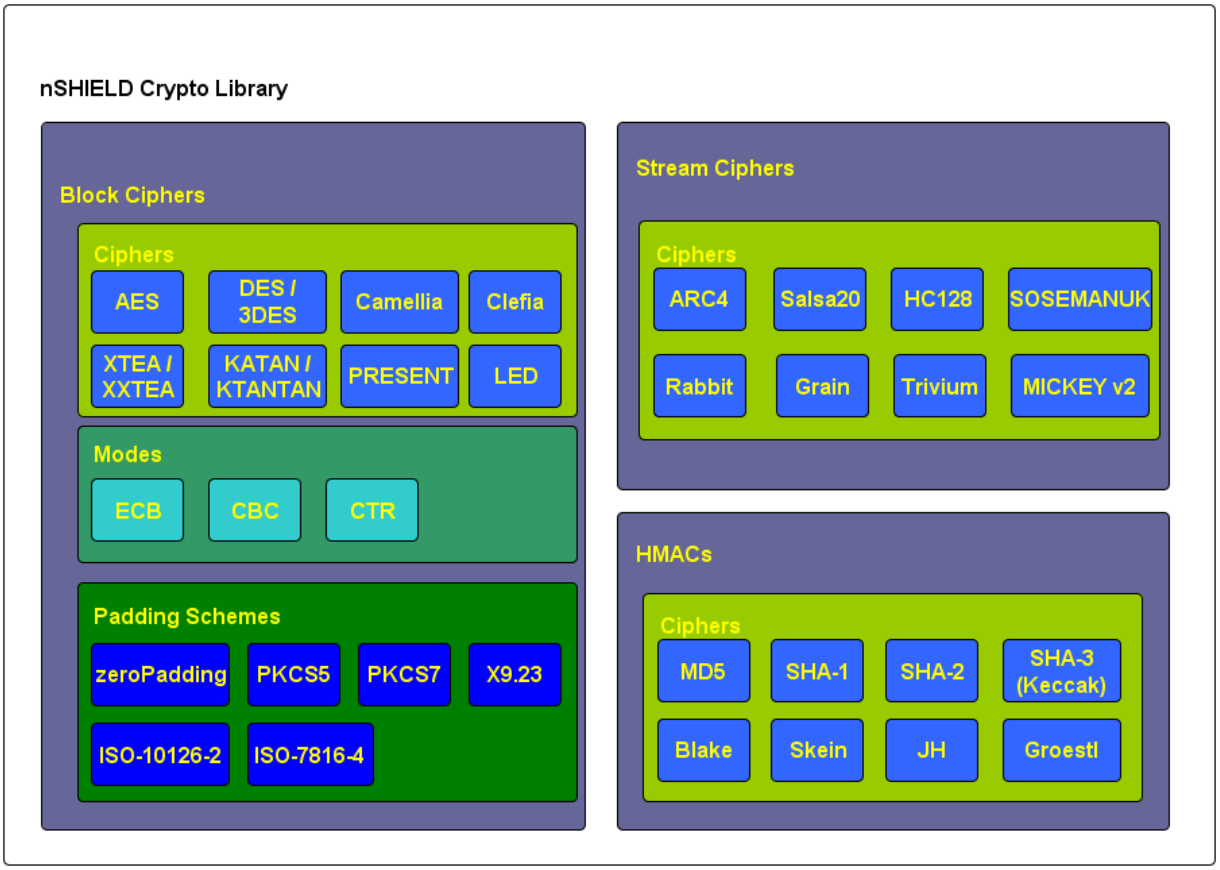


Figure 6-2: Segmented compilation. Each box represents a different compilation option. For example, a user can compile the whole library, the block ciphers only or specific crypto-primitives.

In this first prototype, the library contains block/stream ciphers and hash functions. For block ciphers, it supports AES, DES, 3DES, PRESENT LED, KATAN, KTANTAN, Clefia, Camellia, XTEA and XXTEA. The block ciphers can operate in ECB, CBC and CTR modes of operation. The padding schemes that have implemented are the zeroPadding, PKCS5, PKCS7, ISO_10126-2, ISO_7816-4 and X9.23. For stream ciphers, it supports the ARC4 and the eSTREAM project finalists Salsa20, Rabbit, HC128, SOSEMANUK, Grain, Grain-128, Trivium and MICKEY v2. For hash functions, it supports MD5, SHA-1, SHA2, the new SHA-3 function Keccak and the other finalists of the SHA-3 contest Blake, JH, Groestl and Skein.

6.3 Identity-Based Encryption

6.3.1 Brief description of Identity-Based Encryption (IBE)

Identity-Based Encryption (IBE) was first proposed by Shamir [15] in 1984. It is a type of public-key cryptography that can represent an individual or an organization by a publicly-known string, such as an e-mail address, domain name, etc. It is therefore not required to acquire an identity's public key prior to encryption. A publicly-known identity value in the form of an ASCII string is sent to a trusted third party, called the Private Key Generator (PKG) that generates the corresponding private keys. Upon reception of the ASCII string, the PKG publishes a master public key, which can be used by any party to compute a public key for communication with the entity corresponding to identity ID. A master private key is also computed, but remains within the PKG. For decryption, the party authorized to use the identity ID contacts the PKG, which will use the master private key to generate a private key for identity ID.

Furthermore, IBE potentially removes the problem of trust regarding public keys and introduces trust in the Trusted Authority - TA (a role that may be played by the PKG), which automatically has the role of a key-escrow agency [16]. Namely, if an entity is “expelled” from the system (e.g. a network node that has been misbehaving), its communications can be read, as the TA can use the system-wide master secret to re-compute a private key for any arbitrary identity string, without ever archiving private keys. This greatly simplifies key management, as the secrecy of the base secret need only be protected.

In such an ideal cryptographic system:

- Users need exchange neither symmetric keys nor public keys.
- Public directories (files of public keys or certificates) need not be kept.
- The services of a trusted authority are needed solely during a set-up phase (during which users acquire authentic public system parameters, to be maintained).

As seen in [16] the IBE approach eliminates certificates and the associated processing and management overheads from public-key cryptography. The sender can choose and appropriately manipulate the public key of the intended recipient prior to sending a message has a number of subtle advantages. For instance:

- The sender can provide a preset expiration date for the message by including either the current date or expiration date in the identifier.
- User credentials can be managed more easily, by including the clearance level in the public key.
- Decryption keys can be delegated so that only those with particular responsibilities can read particular messages.

The authors in [17] argue how IBE would seem to be the only practical means of providing security for Wireless Sensor Networks (WSNs). In the research carried out, an implementation of the Tate pairing, dubbed TinyTate, is introduced and the use of IBE to solve the key distribution problem in WSNs is proposed.

In the context of nSHIELD, the role of the PKG can be assigned to power nodes, since some computationally-intensive tasks are involved in the key generation process.

6.3.2 IBE schemes

Several IBE schemes are available, the majority of which is based on bilinear pairings on elliptic curves [18] and their security is based on the Bilinear Diffie-Hellman (BDH) assumption [19].

The Boneh-Franklin IBE scheme (BF-IBE) was the first to have a proof of security, hence its popularity. Sakai and Kasahara presented a new IBE scheme, with a new key extraction algorithm [20]. Chen and Cheng [21] employed a simple version of the Sakai-Kasahara IBE scheme and the Fujisaki-Okamoto transformation [22] to create an efficient IBE scheme.

6.3.3 Boneh–Franklin

This was the first practical and secure IBE scheme. It belongs to the full-domain hash family IBE schemes [23], with their main characteristic being that an identity is mapped to a point on an elliptic curve and is then used for the operations of encryption and decryption. This kind of mapping requires a modular exponentiation, a fairly expensive operation. The calculation of a pairing is also required for the encryption and decryption algorithms, which accounts for a great amount of the needed computations.

The security of the Boneh-Franklin scheme is based on the Bilinear Diffie-Hellman problem (BDHP) and the random oracle model was used to prove that the difficulty for an adversary to cryptanalyse this scheme is as hard as the BDHP. The Boneh-Franklin scheme is resistant to chosen-plaintext attacks and adaptive chosen-identity attacks. Moreover, with the use of the Fujisaki-Okamoto transformation it is also resistant to chosen-ciphertext attacks and adaptive chosen-identity attacks.

6.3.4 Sakai–Kasahara

The Sakai-Kasahara scheme belongs to the family of exponent inversion schemes [23], where a string representing an identity is hashed to an integer that is used in the cryptographic operations. Such schemes are faster than full-domain hash schemes because modular exponentiation is avoided. Its security is based on the q-Bilinear Diffie-Hellman Inverse Problem (q-BDHIP), which is stated as follows:

Given group elements (P1, P2, , , …,), compute ê(P1, P2)1/x.

Hence, assuming that the q-BDHIP problem is sufficiently difficult to solve, then the Sakai-Kasahara scheme must also be sufficiently difficult to cryptanalyse. The scheme is also resistant to chosen-plaintext attacks, adaptive chosen-identity attacks, chosen-ciphertext attacks and adaptive chosen-identity attacks. The security proof of the SK-IBE can be found in [21].

6.4 Secure Cryptographic Key Exchange Using the Controlled Randomness Protocol

In real world applications of cryptographic protocols, the key management problem refers to the life cycle management of cryptographic keys. It includes the necessary operations for key generation; distribution; storage; replacement and exchange; usage; and destruction. In order to retain specific security level, keys used in cryptographic algorithms and protocols must be periodically refreshed i.e., new keys are exchanged between communicating parties and old keys are replaced. The “controlled randomness protocol” (CRP) for cryptographic key management, was proposed as an improvement for the security level of secure communication protocols. The CRP allows multiple keys to be valid at any given time; it neither alters the total number of keys needed in the underlying cryptographic algorithms, nor the need of a control channel to periodically refresh keys. However, the increased security offered by CRP allows for far less frequent key exchanges.

We propose a novel approach of having more than one key at any given time moment. The approach is based on the concept of “controlled randomness” i.e., randomly using keys in a controlled environment. The concept of “controlled randomness” can be utilized in any protocol that uses temporal (ephemeral) keys. It increases protocol security with minimal computational overhead. In the following paragraphs we describe the Controlled Randomness Protocol (CRP).

The concept of controlled randomness i.e., having multiple active keys at any given time moment, offers superior security characteristics compared to conventional protocols. The system designer can reuse well-known cryptographic blocks in a novel way to achieve increased security with minimal hassle:

- minimal computational effort can be induced by CRP in the case that both sender and receiver can maintain a synchronized random number generator.
- the synchronization requirement can be relaxed, if the system can sustain some increased computational effort induced by the KHF (MAC) operations.
- in heavily constrained environments, the two above mechanisms can be replaced by sending the random number j with each packet. In this case, some security is indeed sacrificed since an attacker can know which packet is encrypted under what key. Yet, the intermix of keys allows consecutive packets to be encrypted under different keys and thus, protect against some cryptanalysis attacks.

The CRP allows in all above scenarios to extend the lifetime of each key way beyond the time of a conventional session.

Further, it allows less frequent exchanges of messages in the control channel (if one is implemented), since less keys are needed to achieve a specific security level for a specific timeframe. An attack on the classical key management protocol with a master key of n bits has complexity $O(2_{2n/3})$; an attack on the RNG for the controlled randomness protocol with m keys has complexity O(l2m) (usually for *m*, the period of RNG, it holds m >> n); and an attack on the KHF method has a total complexity of $O(l(2_p + l2_{n/2}))$ where p the size in bits of the MAC keys.

We assess the performance of the Controlled Randomness Protocol when implemented on Beagle Board and Beagle Bone embedded systems. We present our findings from two different embedded platforms: one Beagle Board running Embedded Linux and one Beagle Bone running a custom compiled linux kernel. We provide insights and possible explanations.

7 References

[1] O. V. P. World, “Open Virtual Platforms,” [Online]. Available: www.opvworld.org.

[2] BeagleBoard-xM.org, “BeagleBoard-xM System Reference Manual,” [Online]. Available: http://beagleboard.org/static/BBxMSRM_latest.pdf.

[3] OpenSDR, “Building GNU Radio on the BeagleBoard,,” [Online]. Available: http://www.opensdr.com/node/17.

[4] FreeRTOS(TM), “FreeRTOS real time operating system,” [Online]. Available: http://www.freertos.org/.

[5] International Organization for Standardization, “ISO/IEC 7816 part 1- 15,” [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29257.

[6] M. Jones and P. Viola, “Robust real-time object detection,” ICCV Workshop on Statistical and Computation Theories of Vision, Vancouver, Canada, July 2001.

[7] J. Maydt and R. Lienhart, *An extended set of haar-like features for rapid object detection,* Proc ICIP, pp. 900-903, 2002.

[8] T. F. C. a. C. J. Taylor, «Statistical models of appearance for computer vision.,» Technical report, University of Manchester, Wolfson Image Analysis Unit, Imaging Science and Biomedical Engineering, Manchester M13 9PT, United Kingdom, September 1999.

[9] R. Gross and V. Brajovie, “An Image Preprocessing Algorithm for Illumination Invariant Face Recognitoin,” in *4th International Conference on Audio and Video Based Biometric Person Authentication (pp. 10-18)*, 2003.

[10] T. Ojala, M. Pietikainen and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE T-PAMI 24,* pp. 971 - 987, 2002.

[11] B. Moghaddam, T. Jebara and A. Pentland, “Bayesian face recognition,” *Pattern Recognition,* vol. 33, pp. 1771-1782, 2000.

[12] NVIDIA, “The Cuda® development kit from SECO,” [Online]. Available: http://www.nvidia.com/object/carma-devkit.html.

[13] A. Khoshgozaran, C. Shahabi and H. Shirani-Mehr, “Location Privacy: Going beyond k-Anonymity, Cloaking and Anonymizers,” *Journal of Knowledge and Information Systems,* vol. 26, no. 3, p. 435–465, 2011.

[14] C.-Y. Chow, M. F. Mokbel and T. He, “A Privacy-Preserving Location Monitoring System for Wireless Sensor Networks,” *IEEE Transactions on Mobile Computing,* vol. 10, no. 1, pp. 94-107, January 2011.

[15] A. Shamir, “Identity-based cryptosystems and signature schemes.,” in *In Advances in Cryptology - Crypto '84, Springer-Verlag, LNCS 196*, 1984, pp. 47-53.

[16] S. Srinivasan, “Identity based encryption: Progress and challenges,” *Information Security Technical Report,* vol. 15, no. 1, pp. pp. 33-40, 2010.

[17] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. Lopez and R. Dahab, “TinyTate: Identity-Based Encryption for Sensor Networks,” Cryptology ePrint Archive, Report 2007/020, 2007.

[18] H. J. Silverman, "The arithmetic of elliptic curves", Springer-Verlag, 1986.

[19] D. Franklin and M. Boneh, “Identity-Based Encryption from the Weil Pairing.,” *Proceedings of CRYPTO, LNCS 2139, Springer-Verlag,* pp. pp. 213-229, 2001.

[20] R. Sakai and M. Kasahara, “ID based cryptosystems with pairing on elliptic curve,” *Cryptology ePrint Archive, Report 2003/054,* 2003.

[21] L. Cheng and Z. Chen, “Security proof of Sakai-Kasahara's identity-based encryption scheme,” *In Proceedings of Cryptography and Coding 2005, Springer-Verlag.,* vol. 3796 of LNCS, pp. 442-459, 2005.

[22] E. Fujisaki and T. Okamoto, “Secure integration of asymmetric and symmetric encryption schemes.,” *In Proceedings of Advances in Cryptology - CRYPTO '99, Springer-Verlag.,* vol. LNCS 1666, pp. 535-554, 1999.

[23] L. Martin, "Introduction to Identity Based Encryption", Artech House Publishers, 1 edition, January 2008.