Emulating Trust zone feature in Android emulator by extending QEMU

IV 203X-DEGREE PROJECT

Author

Arun Muthu
muthu@kth.se
861011-1232

Supervisor

Rahim Rahmani
Stockholm University

Stefan Andersson
Sony mobile communication AB

ABSTRACT

The arrival of smart phones has created the new era in communication between users and internet. Smart phone users are able to run their own application along with enterprise applications. In case of personal application, most of them are downloaded from public market, resulting in challenge for the security frame work by threat of losing sensitive user data. So, ARM introduces the virtualization technique in hardware level to prevent the application process completely isolated from the normal world.

However, understanding ARM architecture and internal working is still black box for the user as well as developers. So, in this thesis, by using the qualitative approach like examine the pre research work in open source and ARM trust zone, white paper, internal knowledge from Sony security team, we take a deep look at the architecture of the ARM trust zone in hardware level to analyze and evaluate their implementation. We describe the design and implementation of trust zone features in android emulator with advantages and disadvantages of it in analysis and result phase and conclude with annotation of suitable design on future use to enhance the security domain for secure processing and utility in Android emulator to benefit the user and developer community.

The contribution of this thesis project can be summarized as following: 1) reviewing current practices and theories on implementation of ARM Trust zone; 2) creating a common methodology for handle the research problem; 3) proposing step-by-step approaches by comparing actual working of Trust zone in hardware level with design and idea of emulated one; 4) Analysis and design the appropriate model to solve the research question.

Keywords:

Trust zone, Emulator, Android, Virtualization, and Security

ACKNOWLEDGMENT

I am thankful to my supervisor Mr.Rahim Rahmani for giving me this wonderful opportunity to do Master thesis research under his supervision. Thanks to his valuable experiences being shared with us, the thesis was accomplished in short time period of six months

My heartfelt thanks to Sony mobile communication AB, Lund Sweden for giving me an opportunity to work in their organization

I am really thankful to Mr. Stefan Andersson and Mr. Ben Smeets for their guidance throughout this thesis period. They dedicated their valuable time to provide technical guidance. They gave me confidence and strengthened my faith to perform this research. I would like to thank all Sony – Application security team colleagues for sharing their valuable ideas.

Last, but not least, I would like thank my parents and friends - Dinakaran, Balachandar, Raman, Anitha, Anjani, Mahesh and Ganesh for their kind support throughout my life and endeavors.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

Technology seeking is expanding widely in all corners of the world and Smart phone is one among them. Inability design and improper handling of the security critical functionalities of the Smartphone shows that no technology is resistant toward the security leak or attack till now. Service provider or Smartphone marker adopts the new technology to solve this problem called trust zone [1]. Trust zone is the technology has gained wide acceptance and development in recent times. ARM trust zone is a hardware based system virtualization. This proposal describes the main security and privacy concerns of the ARM trust zone and emulation of trust zone environment with regards to the Android Operating system and extension of QEMU emulator [2]. This research aims to develop knowledge on trust zone and emulate it.

1.1 RESEARCH BACKGROUND

Now a days, we see plenty of smart phone users, progressively increasing in recent years 2011 and 2012 and is expected to be more by the end of 2012 and 2013[8] . Since using smart phone is easy to access the application download, compatible and portable where as in case of laptop and notebook. First android smart phone was introduced in the year 2008. Soon after the release, we found lot of security leaks and vulnerability in the security architecture of the android OS. Since, Android Security Architecture, grants permission to perform any type of operation and So Google proclaimed that, “We tried really hard to secure Android. This is definitely a big bug. The reason why we consider it a large security issue is because root access on the device breaks our application sandbox."[1].

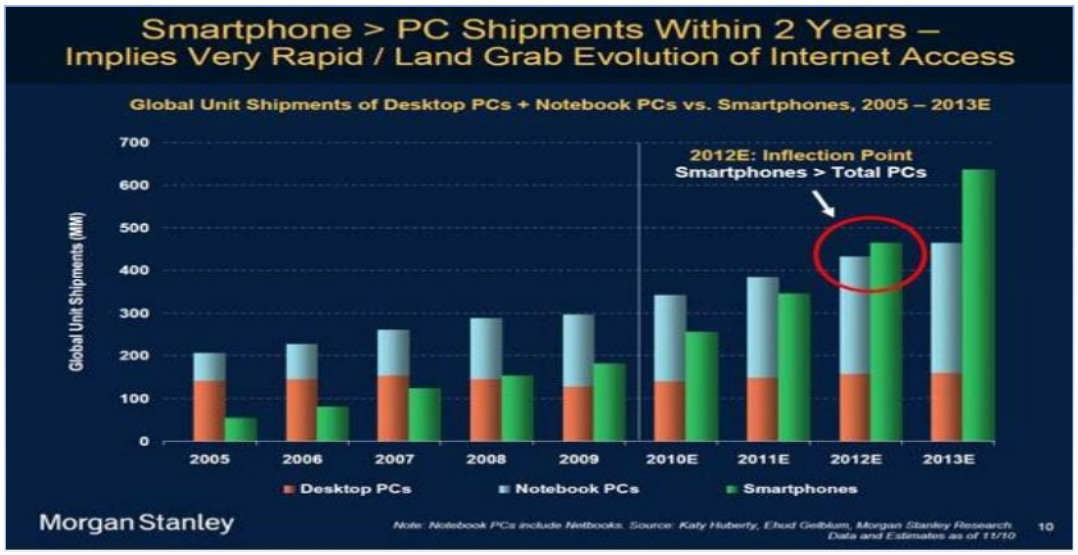


Figure 1 Growth of Smartphone Vs Notebooks and PC’s [8]

ARM trust zone is a hardware based system virtualization. There are two zones in the architecture of the trust zone; they are “Normal” and “Secure” world. The application which requires secure process will enter into secure zone from normal zone. There will be supervisor in the secure zone who will access the data from the normal zone and process it in secured way. Since this process is internally organized [7]. Hence, the User (smart phone) is not aware of all these things. The main functionality of the chip (ARM trust zone) will handle memory management unit, input and output guidance of the data, handling cryptographic keys and certificates etc. Moreover, trust zone will help in handling the third party application and operating system.

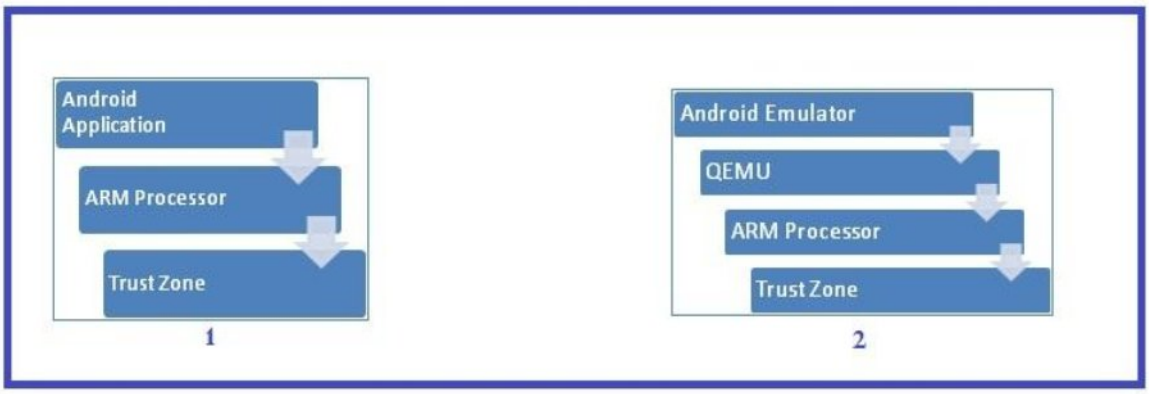


Figure 2 1: Working of trust zone in Target (Smart Phone) 2: Working of trust zone in emulator

This problem comes under purview of Trusted Computing. Trusted computing is a process of virtualization of the operation for specific kind of data, keys, payments, etc. The main way to achieve isolation is trust zone (literally partitioning the memory area between normal and secure process). Till now, hardware is designed for this purpose, complete isolation of process in order to ensure with a secure way with for the treatment to the data [1] [5] [3].

The main purpose of this research is creating the emulation of trust zone in order to reduce the dependencies on the hardware requirement. So that, emulation always is used to forecast, test and minimize the errors or bugs due to security leak [3].

1.2 RESEARCH PROBLEM

Implementing trust zone in android emulator is one of hardest task and probability of getting it achieved is highly reliable. We cannot focus on single model or methodology. Since, we comparing two different entities like real smart phone trust zone with android emulator. There are many practical difficulties in knowing the hardware architecture, programming methods, requirement gathering, time for implementing, future benefits, under-stability, reliability etc.

Different questions to be dealt with at the onset of this project:

- With are all resources available to start this project?
- Has any prior research or reference been carried out for this project?
- Why is this project not achievable from research point of view by others?
- How can a trusted application be created without any dependencies on vendors?

Most of the questions are non answerable or have inadequate information. So defining the scope itself, is a bigger task. Thus the research problem is derived by understanding ARM trust zone. But, for implementing trust zone in an android emulator there exists no standard frame on which to run the application with requiring secure processing.

1.3 RESEARCH QUESTION

Android emulator is helpful for designing the business processes like application, transaction and payment etc. whereas upcoming application (trusted application) for secure process cannot be resolved by the android emulator since there is no support for software virtualization in it. Trust zone is hardware virtualization found in real target (Smartphone), but in case of Android emulator, it is not. Our main goal is to study the architecture of the Android emulator, features of trust zone unit in hardware unit and capturing these two requirements for designing the Android emulator with trust zone.

In simple terms, we are going to study about two things, which directly define the scope of the project:

Q1) How to extend Android emulator to support trust zone feature to develop quality operating secure system?

Q2) How SRAM memory concept will support context switching mode when secure monitor call (secure channeling) is called?

1.4 RESEARCH OBJECTIVE

Aim of the project is to develop the environment; emulate the functionality of the trust zone which works exactly like the ARM Chips which in turns support trust zone. So trust zone functionality can be achieved by classification of requirements. The divisions of Tasks are listed below:

- Allocate the memory (called Secure RAM/SRAM) in the emulator
- Creating Driver file for SRAM memory.
- Swapping of Programs/Application to the SRAM memory area, whenever there is system call for secure processing
- Handling context switching mode between the normal memory and SRAM memory by bank all the required CP15 registers for secure/non secure mode[5]
- Extending feature of SRAM memory by adding Monitor mode.
- Sample Trusted Application

1.5 LIMITATIONS

Since, we are going to develop the system which already exists in real world – Hardware virtualization. Mocking those features in Operating system is huge task. So, scope of the project is huge and vast, even if we try to fix the entire feature of trust zone in software level. Due to time constraint we can limit the scope – to set up a primary feature of trust zone like “Secure Monitor Call” (SMC) and context switching between normal and secure world with help of CP15 registers.

Problem with the special/separate memory is that it is not a protected one. Whenever there is transition between the normal and secure world, there should be some kind of special mode or flag entry in the processor called CP15 register [5].

Creation of trusted application code is another unsolved issue in implementation phase. Since, we don’t have much control on the creating trusted application. It depends on chip vendor as well as Business providers.

1.6 ETHICAL ISSUES

Ethical issues are considered throughout this research, especially during the data collection from Sony internals by signing Non disclosure agreements and other external sources like pre research, white paper, publications etc. The data collected during these processes is used only for academic and research purpose and will not disclosed for any other means. With due respect to rules and regulation of Sony and DSV, information regarding original design, sources, communications are kept secured as per policy and their demands.

CHAPTER 2: EXTENDED BACKGROUND

Trust zone is the security extension of the ARM architecture and protection signal in AMBA3 AXI Bus architecture. The security extension is the concept of secure and non secure state with secure monitor mode. So, new Instruction "SMC" Secure Monitor Mode is added in ARM architecture, which helps us to switch between secure and non secure world, with some additional privileges for secure monitoring. This security extension can be found in hardware level, which does not exist in software or application level (emulator).

The idea of trust zone is having one processor to simulate two worlds: "Secure World" (SW) and "Non Secure World" (NSW), which split the resources by using special hardware specification. For this purpose, a new hardware bit NS an S bit are introduced. If NS=0, then we assume working in Secure world (SW), otherwise it is in non secure world. The bit NS is the output of the processor and is propagated all over the resources, to support separation.

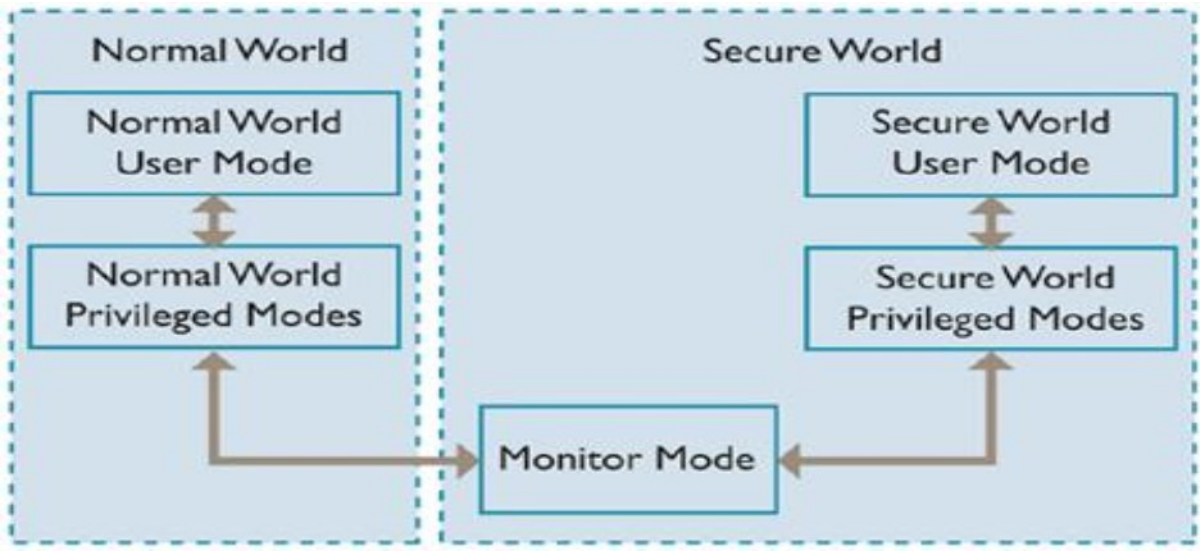


Figure 3 Separation of Normal and Secure world

In reality android emulator uses QEMU (processor emulator), which emulate the ARM processor in the local system. Mocking (sharing memory –mapping) of the real processor (For e.g. Intel processor) as ARM processor. Trust zone software gives minimal secure kernel, which helps to run parallel Rich OS such as linux, Symbian and windows etc. Driver used here to establish the communication between normal and secure world, in order words normal kernel to secure kernel. Trust zone software helps to protect the kernel, peripherals from the program running in normal worlds with help of security extension. Normal world application does not have access to enter into secure world with supervisor privileges. The system without security extensions cannot run the trusted application. Therefore, trust zone software emulation is needed in order to test the trusted application.

2.1 SCRUM

Our research work starts with gathering requirement for designing the trust zone feature in the Android emulator. So, problem statement is clearly defined in the actual theme of this thesis paper. But the question is how to achieve and solve this problem? Any project in software development should obey the principle of Software development life cycle. In this project, requirements are already predefined and also lack in literature review. So, appropriate model or approach has to be chosen. SCRUM is the best model here, which exactly matches our problem design and definition

Scrum is an iterative and incremental agile software development method which is used to manage complex projects & product.

2.1.1 WHY SCRUM?

When a product is developed, its planning, requirement analysis, designing are carried out as initial stages/phases and no changes are allowed once the development process is started [13].

Most of the time there are misassumptions of cost of the project, time estimation and quality of the product i.e. the team always underestimates the span of time required for the project/ resources and hence faces the threat of deadlines! They are always clueless about the amount of work that can be done due to lack of communication between developers and testers and many more.

When developers are nearing the project deadline, there will be tension involved due to unplanned resources, lack in time-to-time review, miscommunication and many more. Hence this brings a need for a technology such as Scrum [13].

Scrum is a methodology which governs the progress of the application development on time-to-time basis with the entire unit working together as one. It involves a lot of brainstorming done by the entire unit. The project is developed based on ideas given by the Scrum master, Product owner and the Development team. Therefore the communication lapse is always avoided and prevents the ambiguity to the developers. This doesn't mean the product is final; there could be some enhancement or changes in the product as well after thorough discussion from time-to-time, where most of the other methods lack.

There are three roles involved in Scrum and they are:

- Scrum master
- Product Owner
- Development Team

The main role of the Scrum master who ensures that the process is followed; the requirements are fulfilled and are taken care by the entire team. So, Scrum master who takes the credit or criticism!

2.1.2 SCRUM VS TRADITIONAL SOFTWARE DEVELOPMENT METHODOLOGY

According to Poppendieck [16], who list out various problem faced in traditional software development methodology like waterfall, spiral, VV model etc

- Documentation for tracking the progress of the project
- Priority of the problem statement or requirement to be achieved
- Lack of communication with client, vendor, customer, team etc
- Waiting for information
- Unwanted development feature of the software requirement

2.1.3 OVERCOME OF SCRUM

"*We are uncovering better ways of developing software by doing it and helping others does it. Through this work we have come to value:*
Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*
That is, while there is value in the items on the right, we value the items on the left more."[17]

Scrum increases the feedback – two way channel. Since it use time track to prevent the unnecessary, unwanted functionalities in development processes. It also increases the commitment within team individually. This indirectly prevents the time bound for commitment of a large team. We can stick to deliverable since goals are clear and time is estimated before the development itself. So, Scrum is called goal oriented approach.

Some more advantages by Poppendieck [16],

- Small team
- Clear cut goal and mission statement
- Fixing the time frame
- Using the proper expertise and experience
- Communication
- Basic environment knowledge

2.2 SOFTWARE DEVELOPMENT IN SECURITY TEAM (SOMC)

2.2.1 WHY SCRUM AT SONY?

This question can be answered from the empirical analysis of the report. Which gives detailed explanations of the problem statement in Sony and the process of scrum was chosen to implement trust zone application with help of company's internal resources and partially with the project manager, team leads, and software developers at Sony.

2.2.2 COMMON SOFTWARE DEVELOPMENT IN SONY

There should be a common approach for method of work carried out in different domains of the organization, here in Sony; the four different teams belong to security domain – DRM, Application Security Team, Flash and Security Team and Project Management Team. This problem can be solved by either finding mutual work flow between the team or create new common work flow. Finding common things of the traditional work flow followed by this team is highly difficult since, they use different platforms within the team itself. However, each work is dependent on one another. So by choosing the common work flow between teams is introduced in order to avoid dependence problem. The conclusion is to create an alternative approach for software development for quick delivery and of high standard.

The approach is to develop a technique or a tool for software development. Since from earlier days, they followed water fall, spiral, Kanban, iterative and RAD approach. But now they are following Scrum tool for software development. The main idea of following Scrum is to find common framework and terms in development stages, which describe what teams are doing during those stages of software development and prioritize of functionalities or problem of the software. This would help teams to communicate even though they work on different platforms without changing the nature of their work.

The common work flow created is represented in Figure 6 as the systems development process. Various drawback are listed out in traditional model, some of them are:

- Waterfall model was very appealing only to the requirement while helping management to control the process in early software development. If the requirement is not freeze or sudden change in the requirement, lead to great disaster. “I believe in this concept, but the implementation described above is risky and invites failure” [18]
- Spiral depends on the amount of iterative is required for the process is very crucial. This will burden the developer who is involved in the process. The development is organized into four phases; inception, elaboration, construction and transition. Each of this phases of software development process, consists of one or more iterations.
- Lean methods are used in manufacturing and product driven companies. Allocating same time on each process in different station is called pull system. Even though lean is more effective than other models comparatively we cannot use it in software development directly as like manufacturing company does. Since time, quality, requirement and mutual existence are important. Team work and dependencies between them are unavoidable.

Due to these factors, development team requires new technique in order to achieve the goal of organization as well as to produce the best quality software product in quick time even relating to vendors.

2.2.3 WHY SCRUM FOR THIS PROJECT?

View about SCRUM over emulating trust and development team

Lean methods which were used in manufacturing have also reached other fields like software development. Any project should focus on lean methodology in order to improve the efficiency and 100% utilization of resources among the team. There are many phases in software development,

To explain this, we need to know the basic information of why lean was introduced? Below figure explain the importance of time utilization and co ordination of different units among the various teams from top management to lower authorities.

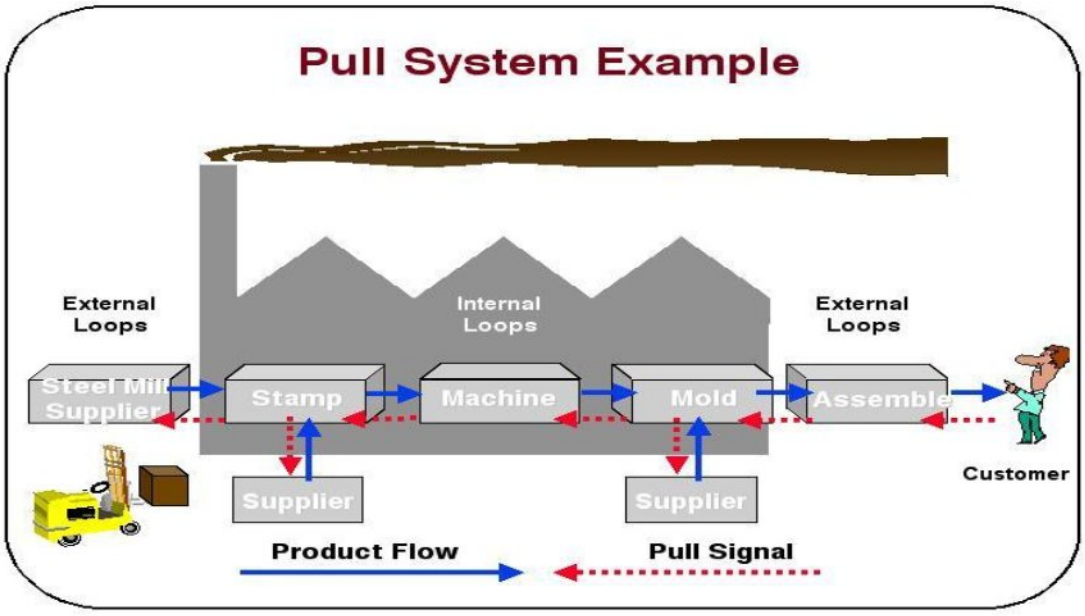


Figure 4 Pull System [19]

Below table, explain the phase of software development using lean method, where as Scrum is mostly used in execution queue and execution phase, because developing the actual feature/product which is important is done here

Definition Queue			Execution Queue	Execution				Done	
Ongoing	Approval	Planning	Ongoing	Development	Testing	QA	Deploy	Feedback	Maintenance

Table 1 SCRUM board

SCRUM is actually used in the software development, Even though entire flow require lean or Kanban, for efficiency. Development phase is quite different from other phases, since we cannot predict the exact output or time. To handle this type of dynamic situation, we need a tool which is flexible to support the needs of the developers.

More or less, importance of development team and their functionalities to be:

- Information back from customer (operators) sends to the development teams to have priorities set based on increased understanding.
- Improved response time for features and issues
- Request customer- development peer to peer dialogue when it is beneficial from development teams to understand the features requested

2.2.4 WHAT ARE BENEFITS OF SCRUM IN THIS PROJECT?

SCRUM is the best tool to define the scope and time frame, to achieve the maximum prior work in this project to make emulator match or work like real smart phone.

"*If we already have working processes, then it needs an improvement without disturbing the system, then SCRUM is the best of tool to handle or solve this issue*".

Since, Emulator has many features matches with real smart phone; the only difference is some hardware specification. So we need to add missing block without violating the existing code or system to add new feature.

"Organization with low knowledge about risk and decision making, and wants improvement with short time frame, then scrum is right choice and best strategy".

Time is the unavoidable factor for any process in software implementation. Quality and reliability are the priorities in this project. Project like emulating trust zone is based on time bound since single person will be handling all this process work. Reporting is best way to follow up the work process. This can be helped by scrum - schedule meeting, informal interviews helps to find the progress of the project/software development.

Advantages listed:-

- Extreme value - reduces risk in ROI
- Supports business value driven software development.
- Control of complex process of product development
- Allows Developers to focus on delivering a usable functionality to the client
- Generates productivity improvements by implementing a framework that empowers teams and thrives on changes
- Insists that the Client prioritize required functionality.
- Ability to respond to the unpredictable in any project requirements

By following the lean board table, we use the same technique to distinguish the problem priorities. In order to use scrum in development phase, we need to know the prior problem, from requirement definition queue phase.

Development phase is divided into smaller problems:

Part one: Creating trust application and Memory handling

Part two: Applying authentication and handling monitor mode and registers

So each work (parts) is feed into sprint like understanding, resource development, coding, testing etc. Weekly basis, information or feedback is announced to supervisor to note the development process.

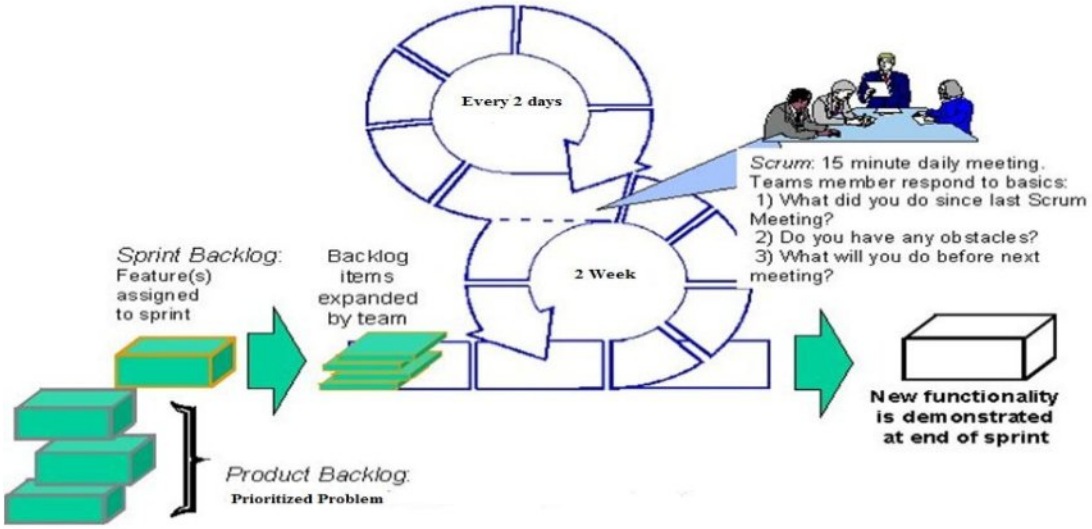


Figure 5 SCRUM [14]

2.3 WORKING OF TRUST ZONE

In this chapter we are going to see about the list of parameters used by ARM trust zone for secure processing.

Parameter required by secure world to run trusted application in ARM Architecture

- Secure memory management
- Monitor Mode and supervisor mode
- Handling registers
- NS and S bits
- Trusted Application
- Secure interrupts
- Debugging

Trust zone software parameters, environment, modes and values are initialized during the booting process. ARM processor triggers the small boot ROM code, to initialize the hardware support parameter in trust zone. This includes, Memory partitioning for both worlds and update the CPU status (supervisor mode and Monitor mode).

The things that happen during the booting process are:

- Instantiate the secure OS with normal booting process
- Setup the MMU(memory management unit) with cache and memory protection
- Set up the mode of each processor and run time environment

2.3.1 SECURE MEMORY MANAGEMENT

The theory of secure memory management is the division of the physical memory into secure and non secure area, which in order improves the integrity of the data. The secure monitor mode will ensure the grant access to only secure world process, memory and peripherals. The partitions are wired and configurable. A bus controller signal is used to guide the navigation flow or access between the worlds according to NS bit.

When application requires secure processing, NS bit changed to 0 and S bit to 1, to intimate the CPU that application requires secure processing. The application is then moved to secure region, where normal physical access is denied. The below figure explain the secure memory management in ARM architecture.

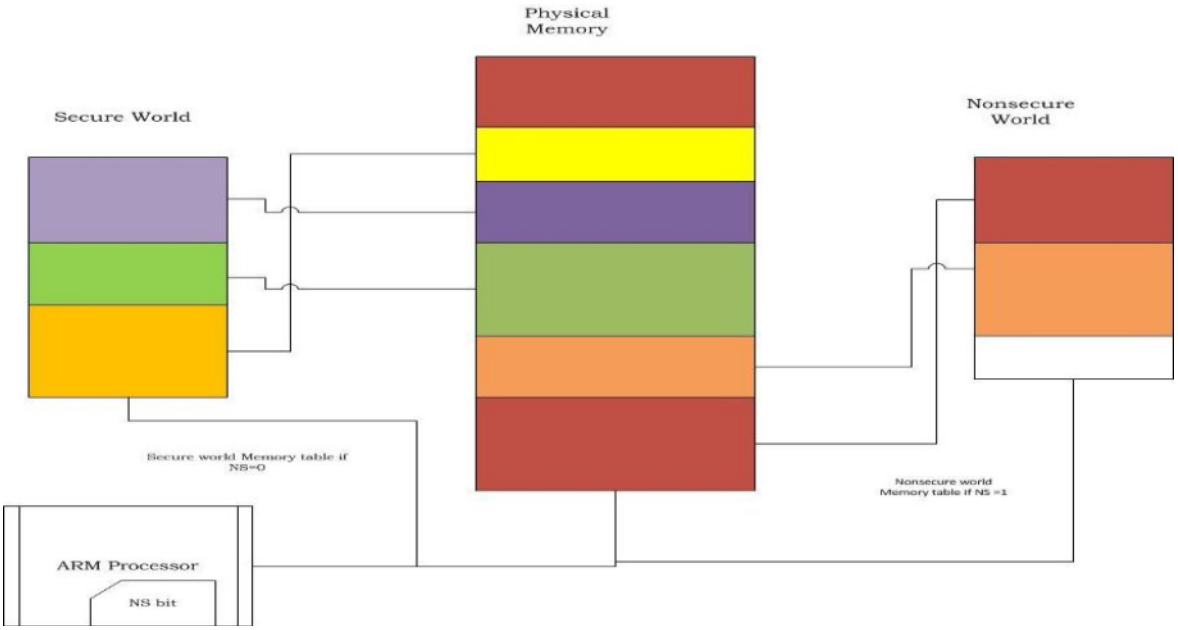


Figure 6 Separation of resource and its utilization in ARM –hardware

2.3.2 SECURE MONITOR MODE AND SUPERVISOR MODE:

During the transfer of data from normal memory region to secure memory region, we need to do two things. They are

(i) check for authenticity of the code, i.e. whether application is trusted one or not

(ii) If application is trusted, then change the mode of the bit and cp15 register.

	Monitor Mode	Supervisor Mode	Non Secure bit	Secure bit
Application in Non secure region	0	1	1	0
Application in Secure region	1	1	0	1

Table 2 Bits change value for secure and non secure

Non secure bit and secure bit only determines program is running in secure or non secure world. The bit values are stored in coprocessor called CP15 register "Secure Configuration Register (SCR)".This register values can be used for both secure and non secure user modes and privileged modes. In order to access SCR, we should be in secure monitor mode =1, otherwise it leads to exception. Monitor mode change is the actual gateway to and from secure world (secure memory region).

It takes the current state of the processor and saves at some a secure place (in register). Afterwards the monitor takes a previously saved state of the new world and restores it. The final step is to set the NS and S value and to delegate the running time to code of the new world. This mode change can guide from Secure Monitor Call (SMC), An ARM instruction use to change the secure monitor mode and perform secure kernel service layer functionalities.

2.3.3 INTERRUPTS

ARM architecture support two types of interrupts namely IRQs and FIQs. IRQs are handled in the normal world, but FIQs can be handled in both normal and secure world. If the processor is running in the secure world and IRQ arrives, then the processor calls monitor automatically. Monitor switches contexts and then the IRQs handler runs. After the interrupt is finished, the processor stays in the normal world continuing the process that was previously paused in the normal world. The same mode of functioning applies in other direction.

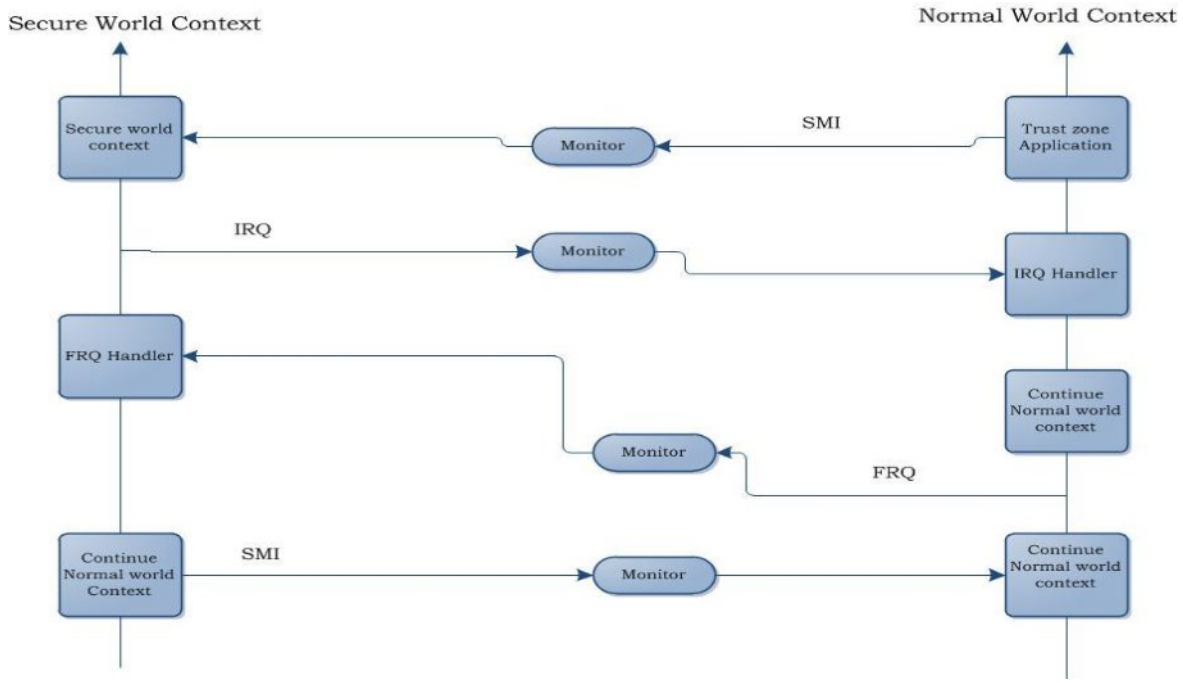


Figure 7 Interrupt Handling

2.3.4 TRUSTED APPLICATION

Trusted application is like any other application, which runs in secure memory region. If it is loading it in secure memory location, we need to find integrity of the application either by certificate or signature.

Figure 8 Trusted Application

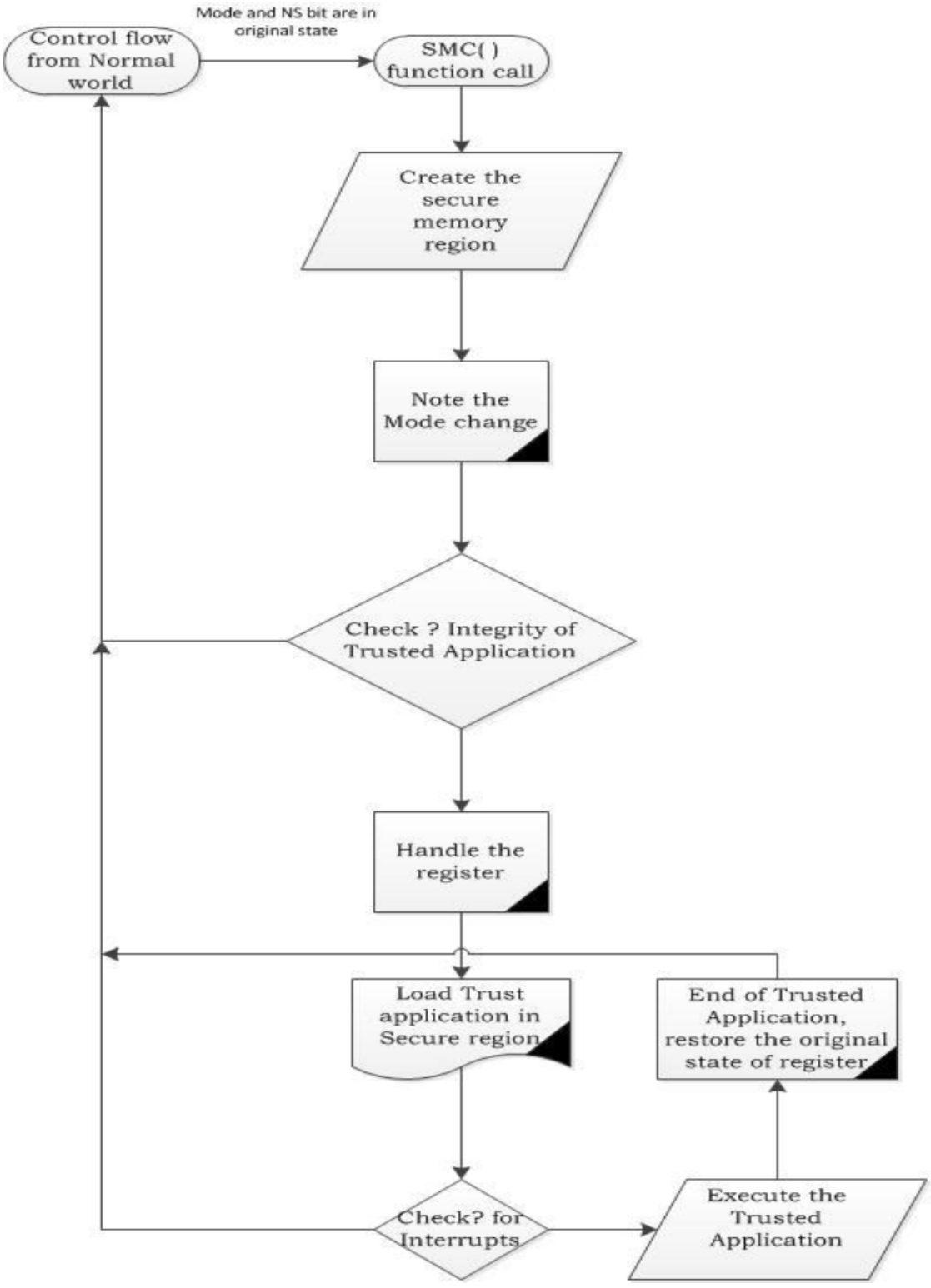


Figure 9 Overall control flow

CHAPTER 3: RESEARCH METHODOLOGY

This chapter discuss about the research approach, data collection, strategy and literature to supporting the research question and area and to identify the limitations and problems in executing this project in real time.

In order to solve this problem, we need to follow scientific perspectives, called Positivism and Hermeneutics. The aims is to create intellectual capacity knowledge via construal and analysis, we finalize hermeneutics is the most appropriate approach to explain the interpretation analyses and meaningful logical concepts [6].

Figure 10 Underlying philosophical assumptions based on Myers [9]

According to Ricoeur without structural understanding it is impossible to gain interpretation knowledge [11].This statement is suits to our problem. Moreover, our goal is software product as the result, so we are bound to follow Software development life cycle (SDLC). In order to study this, we are having two methods, Qualitative and Quantitative.

"*Qualitative research would define the being of fishing, the ambience of a city, the mood of a citizen, or the unifying tradition of a group*" (Boris Blumberg Et al 2005, p 192)

The legitimacy of the research can be amplified by using Triangulation. Triangulation is the process of the using several methods to understand or examine the problem statement or topic. According to Björklund & Paulsson [12], there are method and data triangulation. We use method triangulation, since we are going through literature reviews, reports and observations etc. we also use data triangulation, to collect the data as form of semi formal interviews with pre researchers, project leads, senior engineers, software developer etc [Appendix III].

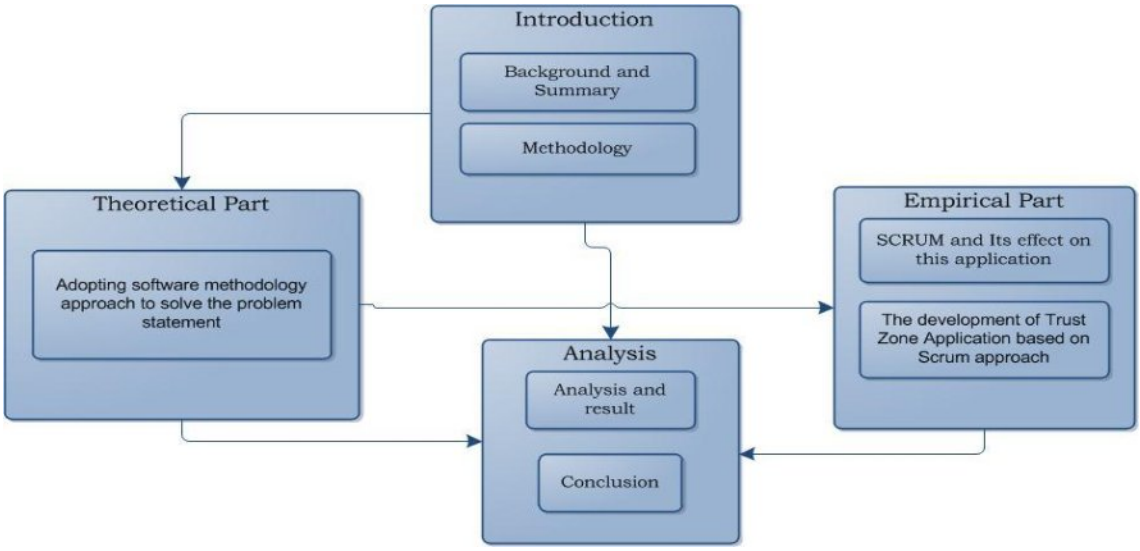


Figure 11 Structure of this thesis

The process of practical work is divided into phases shown in the above figure 11. We begin this process by understanding our mission, via unstructured or semi formal interview with our supervisor and project manager to ascertain our understanding over the goal and mission of the project. Then started to write the abstract and theoretical column and with help of interview we can also write the empirical column, later we use these information was used for our analysis. Since, this research include verdict of factors of trust we chose qualitative methodology.

Quantitative studies are based on the number and figures and Qualitative studies are based on valid and trusted information like words and sentences. So, Qualitative studies will be more suitable for problem statement. Since, most of the knowledge for this research gained from literature review.

We are comparing the real work of trust zone features from smart phone and integrating these features to emulator. Emulating trust zone in emulator is the process of replicating the hardware Functionalities into software one. Traditional software approach like waterfall, spiral etc. won't fit into the software development. Since already specification and requirement are defined in the Hardware trust zone design phase. So, we need to follow the new software development methodology approach to fit into the problem definition. For that, SCRUM is the best approach to solve this problem.

3.1 OBSERVATION OF WORK PROCESS

First step in establishing the knowledge for our topic or problem statement is via literature and studies. This conceptual study always leads to understand the entire work process, and evaluate the empirical data in opposition to our theoretical interpretation of relationship involved with software development as centre of focus to carry out the work process. We have frequently reviewed the source to concentrate and gain more knowledge and foundation for interpreting our observed data in a working process.

According to Donald R. Cooper [22], semi formal interviews with stockholders, will help us in designing process. So we can gather required information/suggestion, which benefits the stockholders. Reviews report, code review, small meeting will help to improvise of the project design.

For Question Q1, *literature review will be helpful; these details can be obtained by following SCRUM approach.*

- Literature Review
- White paper, research papers, books based on trust zone
- ARM Architecture trust zone
- Study about Android programming
- Linux drivers
- Examine QEMU Emulator (git-hub, linero)
- Examine the code of trust zone, in real target device.

For Question Q2, *Semi formal interview are used*.

Semi formal interviews are conducted among various team of Sony mobile, to define the scope and benefits of the project. So, we focus on functionalities of DRM team, security team, flash and security team, Hardware specification teams, ST Ericsson AB. Discussion with managers; teams and project leads will be done in order determine the scope of the project or outcome. The results/suggestion from the teams will be used to implement the feature of trust zone in Android emulator in code level.

When we started studying the problem statement and goal, we have huge sources related to it. To solve the same problem, there are several ways like Kanban, Agile, RAD, Traditional software methodology, scrum etc., but we must be very keen in choosing appropriate

methodology to support our goal and problem statement. According to Björklund and Paulsson [12], gathered information in the literature, which are presented for questioning both the information and material used to collect it.

According to Leth and Thurén [15], extension of source is must to analysis criteria for assessment of guidance. The four criteria are time, authenticity, dependence and bias.

Time: keep on tracking the process.

Authenticity: Semi formal interview with the project leads, scrum master, technical head always keep in track with problem statement

Dependence: this research should motivate future enhancement and pre research work should be properly handled

Bias: Confidential information like design, architecture and source of the information should be considered as fact and explanation.

These are factors which help to understand the background of the problem and also provide the knowledge to analyze factors in theoretical and empirical way.

In order to add knowledge from literature review, we need to make observation. According Björklund and Paulsson [12], there are two type of observation.

1. **Act like outsider**, i.e. without taking part in the activity or process

2. **Act like insider**, i.e. active participant in the process.

In our case, we choose second one since working with live team to gather as much information as possible. We are acting like observer to note down key points and at end of session or meeting we recollect all key points to prepare a summary. These functionalities help us to understand how real applications of organization are developed within the team. We have noted the observation like deadlines, priorities of the modules in Scrum chat. Meeting with team leads and manager held up every week to discuss progress of the project and deliverables for next coming weeks.

3.2 RESEARCH STRATEGY

The Main way to gather information by qualitative methodology is either by conducting interview or survey. In our case, Interviews will be the best option since we need to consider the importance of people like project manager, supervisor and software architect etc and their time. According to Patel and Davidsson [20], face to face interview is much needed for specific information like confidential data, design and architecture of the software etc.

Meeting observation are gather as single summary and used as input for further work process and also to discuss position of process according to activity plan. According to Björklund and Paulsson [12], the main drawback in this process like conducting face to face interview, we have plenty of interrupts either interviewees can misguide us or else motive of the question may be unclear to the interviewees because of which unrelated answer is delivered for the question. So there is always change in the priority of the action to taken.

Another drawback can be that some of the answers might have been misinterpreted by us. Sometimes interview may be anonymous (since to discuss the idea of the design of trust zone is not clearly known). So, after collecting the views, they are fed into scrum board for analyses. Moreover, system we are going to develop is already existing real world – Hardware virtualization. Operating system which handles this problem is open source. So, scope of the project is huge and vast, if we try to fix the entire feature of trust zone in software level. Due to time constraint we can limit the scope – to set up a primary feature of trust zone like “Secure Monitor Call” (SMC) and context switching between normal and secure world with help of CP15 registers.

3.3ANALYZING METHODOLOGY

First we need to understand, how to convert the summary of the observation (information gathered from semi formal interview) into analyzed result or answer. We can see many similarities and difference from the interviews conducted and categorized according to perception, underlying the structure of the problem statement. This is called Phenomenographic analysis by Patel and Davidsson [20]. Conducting the interview is routine process, still the end of the thesis we need to conduct the semi formal interviews with others like project manager, supervisor, software developers and architect etc.

In order to move further, we need to understand the pattern tied with the problem statement. These patterns are formed by division of problem statement and interviews conduct. Analyze of the problem which depends on the structure of the pattern of data we collected. This process is iterative one; need frequent change in goals, decision making, structure, development process and factors related to it. These types of rapid change in goal and problem statements are not easy to carry out in practical aspect as we mentioned in theoretical aspects. By this way, Scrum came into act for understanding phenomenographic approach for continues progress of the result.

Since we use inductive and abductive reasoning for producing pattern with methodology and phenomenographic approach, we feel Scrum is the best approach for our research. Exploratory study is requiring on scrum, since the problem is not clearly defined. To develop an information system, we will navigate via different phases of the design process to achieve specific problem areas. Since this paper deal with future technology pre research work is very much limited. So we must relate different theories and software approach for empirical study of project. In order to support the problem areas, previous work in scrum along with Android application project are taken as comparing perspective with our project to define the set of pattern for empirical study [10].

CHAPTER 4: ANALYSIS

4.1 GENERAL ANALYSIS

Android (vs.) other operating system

Almost all smart phones available in market run either on one of the following operating system i.e., android, windows, blackberry, IOS etc., ARM support all these operating system to run on it. There are many factors to support android operating system compared with other operating system mentioned above. Android operating system is open source, which directly ensures all the changes and, updating to the operating system without considering the rational assets. This nature motivates the people/developers to work with systems and share the resource and knowledge. This will be tremendous act for the future developers and researchers.

Scratch (vs.) pre-research

Since Android is an open source, so there are many pre research works had been done. For example Johannes winter works on QEMU on Fiasco microkernel and github are the best way to start this project not from the scratch.

ARM trust zone (vs.) Emulated Architecture

	ARM trust zone	Emulated trust zone features
Secure Memory Management	Created during booting time	Created during execution after SCM call
Monitor mode	NS and S bit value changes according to transit between the worlds	NS and S bit value changes according to transit between the worlds
trusted application	Loaded during boot time and it's not dynamic	Loaded during execution time and its dynamic
Interrupts	Operating system take care of it	Operating system take care of it
SCM call	ARM instruction	Procedure call
Debugging	We cannot debug application since its already compiled	Here, Native C code in JNI layer user" *.so" file
Registers	ARM registers	Variables

Table 3 Difference between ARM and Emulator Architecture

4.2 ALTERNATE DESIGN ANALYSIS

Supervisor Design

In this design, we use two instance of kernel layer. Each one represents the one world (normal and secure). Supervisor is responsible for the analyzing the instruction and then send them into appropriate world. This shows that, supervisor is keen responsible for context switching between the worlds. The supervisor design is more efficient since it differentiate the two kernels layer and therefore the two worlds. However implementing this solution is more difficult, since one software module will need to understand the already existing and compiled code.

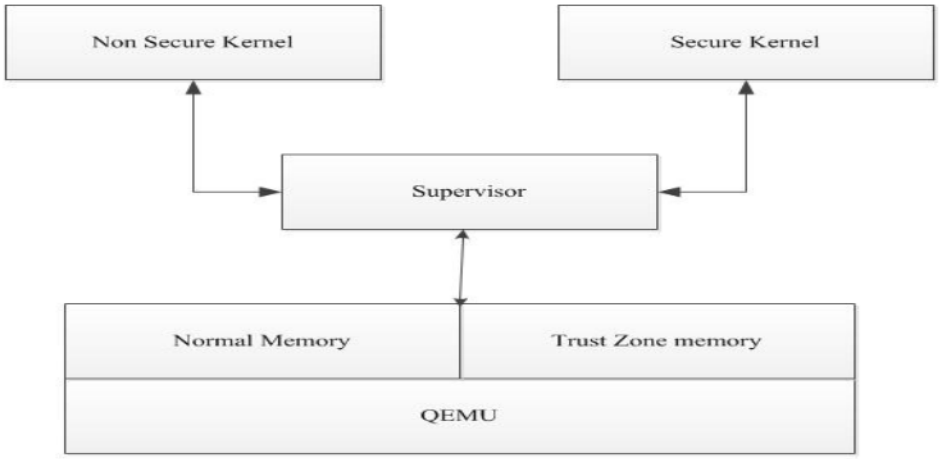


Figure 12 Supervisor design

Dual memory Design

In this design, we create two memory management units to manage each kernel layers to support previous design (supervisor design). As like previous design, here two memory units represent two worlds. The secure memory unit is responsible for the context switching between the worlds. The normal memory unit, will access the part of secure memory unit in order to convey the results between them. So, this indirectly leads to creation of the shared memory region.

This design is not so efficient to isolate the two worlds, even though it solve the supervisor design problem i.e. creation of software module to the stack. Handling memory unit make this implementation of design so complex.

The design is described in the picture given below:

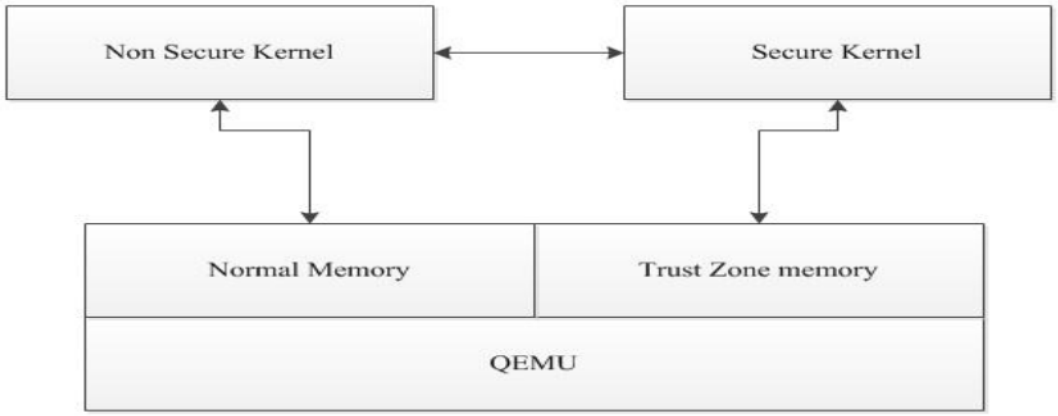


Figure 13 Dual Memory design

Static memory Design

By seeing all these complexity in the nature of design, we will stick to one kernel and memory unit. Isolation of secure world can be achieved by creation of static memory in the section of physical memory. The Kernel will monitor each instruction and will forward it according to the appropriate memory region. The kernel is responsible for the context switching. The secure world can access the data from both world and communicate results to normal world without any shared memory concepts.

This design is much easier comparatively to other design mentioned above. However, the main problem is handling security features i.e. making static memory region has secured one.

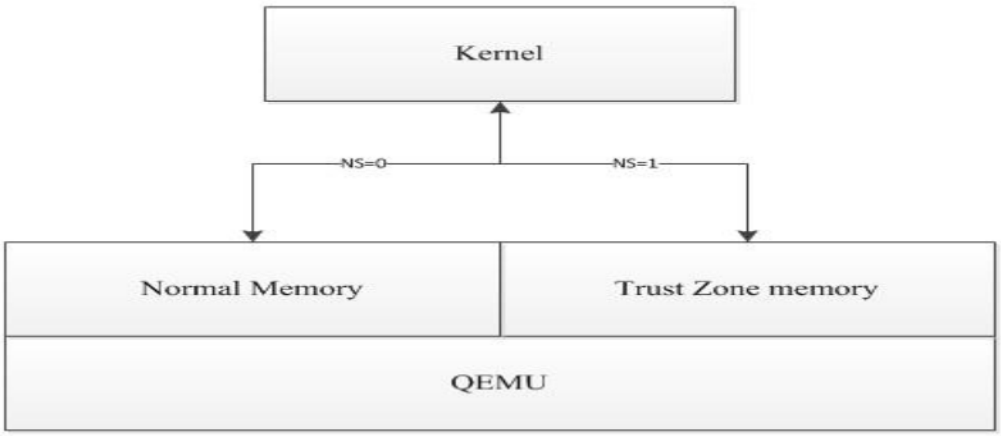


Figure 14 Static memory design

Trusted application Analysis

Actual design of the trusted application is originated from Nordea banking transaction mechanism .They use the challenge response mechanism to authenticate the user to access the appropriate account. This idea motivated me to use challenge response mechanism to authenticate the host with other external entity and to establish the secure channeling between them.

Challenge response between emulator and external world is designed based on RFC 5802, called "**Salted Challenge Response Authentication Mechanism**" [21]. Here in RFC 5802, password is generated using pseudo random number generator which is stored in database, which leads to offline dictionary or brute-force attack. So, it uses salted and hashing technique to ensure more security and authentication exchange.

"*The amount of time necessary for this attack depends on the cryptographic hash function selected the strength of the password, and the iteration count supplied by the server. An external security layer with strong encryption will prevent this attack*" [21].

Challenge response never stands for full secure processing, it just a mutual authentication mechanism. To ensure secure processing, we use pseudo random number generator to create cryptogram and encrypt it with some encryption standard. So, these designs help in designing trusted application layer in our emulator.

To establish secure channeling:

- The client (Host) and Server (External entity) should aware of protocol design to avoid incompatibilities problem between them
- The server starts the key exchange with client to provide authentication
- The client uses his relevant cryptogram to authenticate the server on connection
- Since cryptograms are encrypted using encryption standard (eg: DES). So, key used there is treated as symmetric session key, a secure connection is established to exchange data between client and server.
- When session ends, session key will be expired, for new session, new key will generated

Difference between RFC 5802 and Secure Channeling

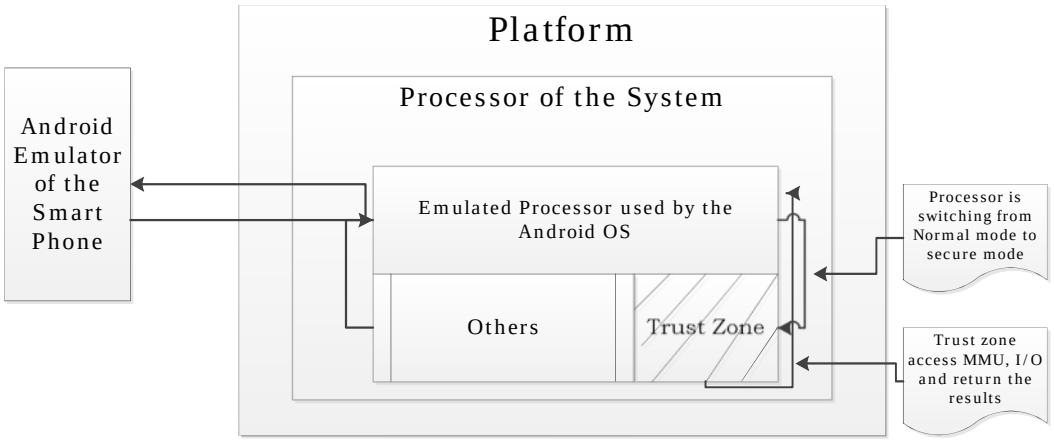
RFC 5802	Secure channeling
Salted Mechanism is used	No Salted mechanism ,Since we are not storing and retrieving any data
Hashing is used to create the “Specific Key” : HMAC(Stored Key, "GSS-API session key" || Client Key || AuthMessage)	Instead of Hashing, we encrypt the derivation challenge data with DES to form “Session Key”
Pseudo random generator is used to generate the password	Here also, pseudo random generator is used to generate the cryptograms
No Encryption standard is used	Encryption standard is used eg : DES, RSA,3DES etc

Table 4 Difference between RFC and Secure channeling

We implement an Android application that will use the TPM services running in the secure world to encrypt a small amount of sensitive data specifically cryptograms. The application will be able to access the encryption standards generated by our DES key generation program to encrypt data and will be able to use decryption keys to decrypt data. This application will run on the secure world.

CHAPTER 5: RESULT

Software emulation of trust zone in other words, moving hardware trust zone into software implementation. Following figure shows how the trust zone is added to processor of android emulator. So that, switching between the normal and secure process can be easily emulated and studied easily.



Pictorial representation of Trust Zone implementation in Emulated processor

Figure 15 Trust zone implementation in emulated processor

Things considered during the implementation of trust zone feature in emulator:

The project is divided into two phase:

- Configuration
- Implementation

5.1 CONFIGURATION OF TRUST ZONE IN EMULATOR

QEMU is a machine emulator to simulate many number of processor architectures such as ARM, x86 and even more. QEMU is used in simulation of application–level code with support from Linux kernel. There are many patches for available in QEMU source tree (git-hub) which aims to add more complete support for ARM trust zone. The code used for the configuration can be downloaded from the git-hub website [4]

There are two major file CPU.h and helper.h mainly used in this section for configuration:

CPU.h

- arm feature to the emulator

```
enum arm_features {
        ARM_FEATURE_TRUSTZONE, /* trust zone Security Extensions. */
};
```

- CP15 register

```
struct {
        uint32_t c0_cpuid;
```

```
    uint32_t c0_cachetype;
    uint32_t c0_ccsid [16]; /* Cache size.  */
    uint32_t c0_clid; /* Cache level.  */
    uint32_t c0_cssel; /* Cache size selection.  */
    uint32_t c0_c1 [8]; /* Feature registers.  */
    uint32_t c0_c2 [8]; /* Instruction set registers.  */
    uint32_t c1_sys; /* System control register.  */
    uint32_t c1_secfg; /* Secure configuration register. */
    uint32_t c12_vbar; /* secure/nonsecure vector base address register. */
    uint32_t c12_mvbar; /* monitor vector base address register. */

} cp15;
```

Helper.h

- This is in charge of Page size of trusted application.

In the remainder of this section, after a brief introduction to the basics of the ARM architecture and QEMU's internals, we discuss the implementation challenges, details and current limitations of our trust zone implementation for QEMU.

5.2 IMPLEMENTATION OF TRUST ZONE IN EMULATOR

There are two types of implementing emulation:

- Top level emulation (above the driver)
- Low level emulation (below the driver)

Explanations about these two of emulation are as follow:

1. We continue with the SMC emulation track which gives us a clean/empty secure world that we have to fill with something and basically do a own TEE (e.g. Global Platform TEE)

2. We skip the SMC emulation and does the emulation one step higher at basically the TZ driver level where we can load a TZ application and start exchange data with it. We thus replace the actual TZ driver with the one that is used in the emulator only.

5.2.1 EXPLANATIONS OF SCM CALL ON BOTH LEVELS OF EMULATION:

SCM is the ARM instruction, which is in built in the processor. Whenever secure processing is required, SCM (# addr) will be invoked in the assemble code of the driver. Trusted code can be deployed in two different ways. One way is to include the trusted code image in the Boot ROM. And other is including it in trusted application which is stored in file system of the Operating system.

Idea proposed after the some set of detailed analysis is as follow:

When we try to emulate this instruction – we need to replace it with the suitable function (system call) which is targeting the particular address location of the trust application (normal C function call) in the file system.

Problems in Low level emulation:

> This is not achievable, due to the compatibility problem between the ARM (phone) and Intel (host machine).In real phone there are two different memory unit and accessing point to prevent and guide the execution flow, where as in emulator, single memory strip and single mode operation.

> The trusted code is split into two parts, when loaded by the boot ROM. The first part is regarded as trusted application persistent in secure SRAM during the control flow. The other part is initialized or triggered from the Top level – Application (android).This indirect leads to undefined problem, that is trusted code cannot be unloaded or reloaded.

> Trusted application (TA) uses the functionality of the Secure ROM API. Secure ROM API is the main API to be used by trust application (TA). So, it is not possible for dynamic trusted application to use defined functionality. So TA should be always static and specifications are loaded in boot time itself.

> Due to this, some prerequisite condition – Boot ROM (system on chip) are required, if we try to use the predefine TA and trust zone driver. This is problem when we consider in terms of emulator. Since, it doesn't correspond to the real CPU.

> If we try to use the existing driver file with our own TA (dynamic TA). We cannot establish the communication between the secure world and normal world. For example: Accessing global variable is difficult. No firmware is there to execute it.

> In handling environment: In git-hub, there are two files CPU.h and helper.h in Target arm, have some code for handling dynamic TA(created by our self) thus leads to undefined error and we are unable to build or run the application.

Problems in Top level emulation:

>Main problem is to create our own execution flow – from Top to bottom – Android application, Trust zone Driver, SCM mocking code and trusted application.

>We need to create the fake monitor code – i.e. initialization of trust application and its properties (key, ID, Data, flag and address etc) is mocking the boot ROM specification.

>Developing entire Qemu environment is time consuming. We assume and designing our own apk and TA to perform secure world operation. Handling register functionalities are difficult since no CP15 register for secure and non secure banking.

>There is a contradiction when we create certificate and signature for Apk and verified before the calling driver itself. This indicates that TA is loaded dynamically. This issues lead to problem in granting permissions to the user.

> Since coping the TA to specify address has no meaning in it. However we need to bring realistic view of ARM processor.

> No details about secure ROM API: so we need to create monitor mode as just variable or flag status to indicate the user or programmer that, CPU mode has been changed and control flow is switched to secure processing.

Basic architecture of the trust zone Implementation along SCM [Appendix III]

By seeing these entire problems, we prefer, Top level approach is more relevant in this case: Explanations for the above is as follows

> working as standalone program

> debugging is quite easy

> Navigation from APK->Driver->SCM_CALL->TA is notable.

> We can protect the memory region (allocate secure memory we created) using mmap methods.

5.2.2 SECURE MEMORY MANAGEMENT IN EMULATOR

We followed the same principle and design to develop the trust zone features in an emulator. But there is slight modification in the design, when we consider the same approach in emulator. Since, there is no special hardware support for the working flow. Since, in real target there is concept called shared memory concept between the processor, but in emulator it is missing. So, we redesign the resource utilization as per emulator accordingly with same NS bit.

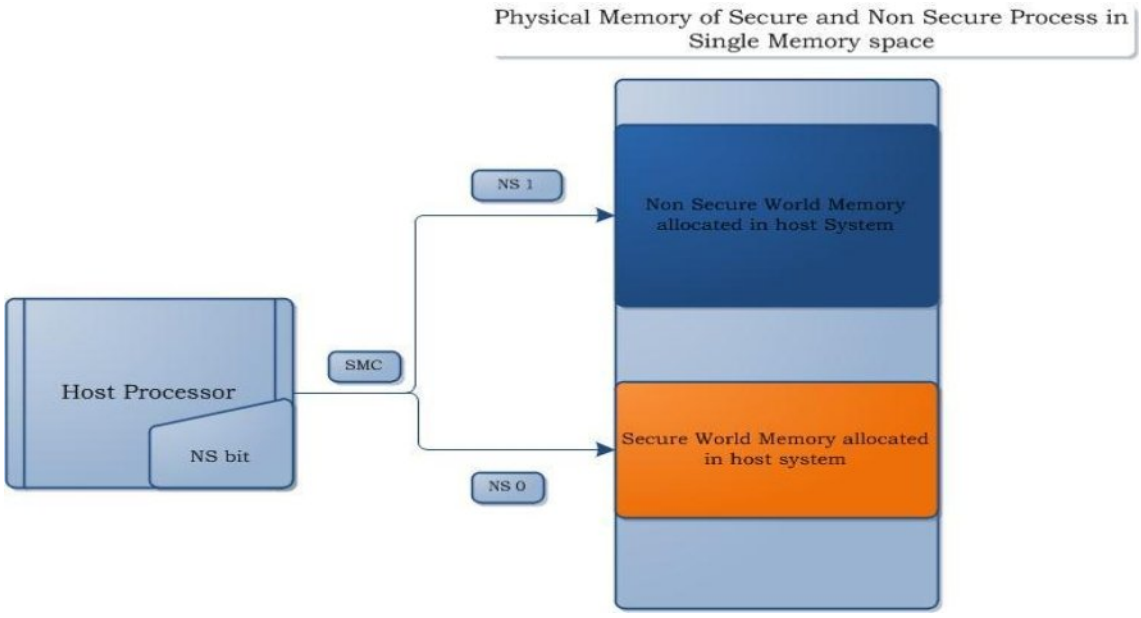


Figure 16 Separation of resource and its utilization in Emulator

5.2.3 SECURE MONITOR MODE AND SUPERVISOR MODE

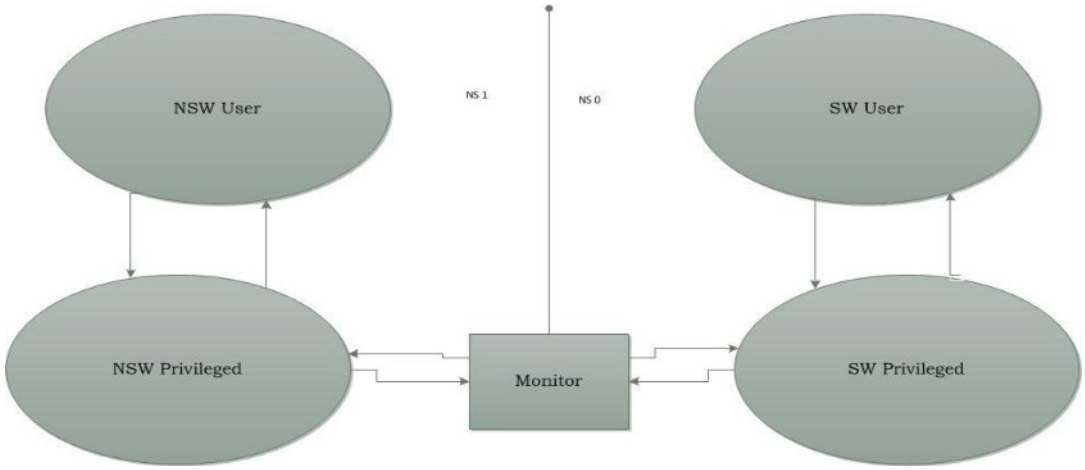


Figure 17 Secure Monitor call

In case of emulator, SMC can be replaced by the function or procedure call. Since, SMC is ARM instruction, which will not work in emulator. SMC function call helps to prevent the non secure state in accessing region of physical memory, since each states operates on own memory address space. Mode changes are captured by the variable values in the programming logic and tracking them are also be done.

The general purpose registers and processor status register are not blanked between the secure and the non secure states. When execution switches between the non secure and secure states, ARM expects that the values of these registers are switched by a kernel running mostly in monitor mode. Whereas, system coprocessor register are banked between the secure and non secure security states. A banked copy of a register applies only to execution in the appropriate security state.

5.2.4 INTERRUPTS

If an exception is taken in monitor mode in non secure state, the secure configuration register bit is set to zero. These forces secure state entry for all exceptions. However, if an exception is taken in monitor mode in secure state then, register bit is not set to zero.

Many uses of the security extension can be simplified if the system is designed so that exceptions cannot be taken in monitor mode. Setting bits in the secure configuration register causes one or more of external aborts, IRQs and FIQs to be handled in monitor mode.

5.2.5 USER SPACE

User space is the main feature of the design, since the control flow start from it. It requires the delegate operation to an authorized domain. The applications have specific system call to the operating system kernel. This include syscall numbers, interfaces etc. In general, environment for secure world user space application should be simple system call interface to support C run time libraries and compiler tool chains.

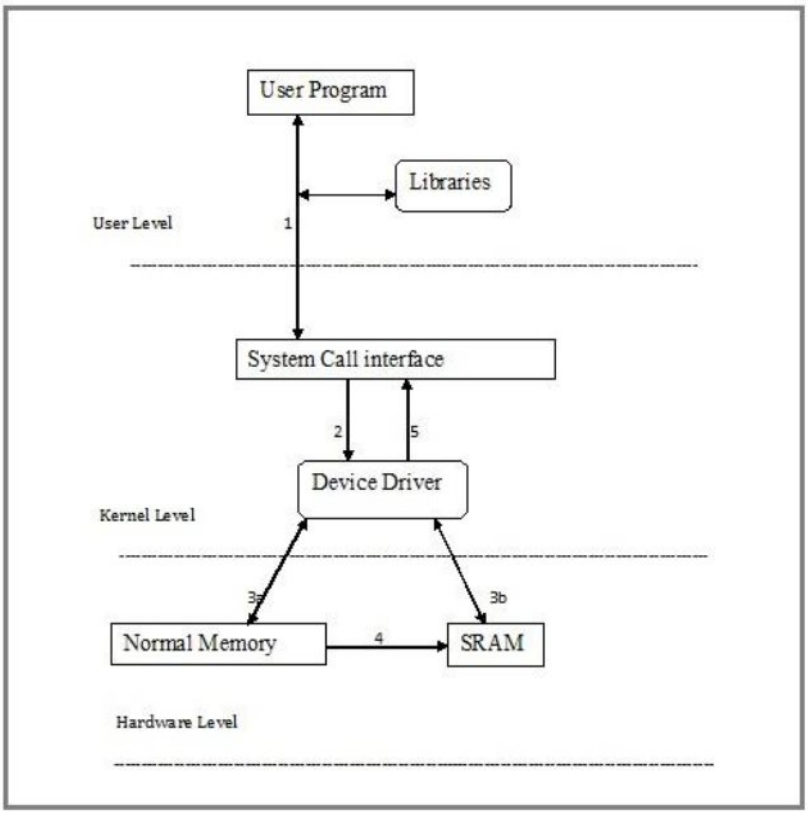


Figure 18 Implementation of trust zone into Emulator by adding SRAM Concept

1) Application enter into the user level (android *.apk to system interface)

2) System call leads to corresponding driver file

These are things are connected by the JNI- Java native interface

3a) initially all process start in normal memory

3b)whenever the special treatment is required during the execution, control will directed to SRAM via driver file access (implementation of methods in tee_sram_driver will do the corresponding action like session creation, pointer reference to data, capturing the mode of input data). We can include some lib file for cryptographic operation, arithmetic operation etc

4) Transfer the block of data from normal memory to SRAM memory which require the special secure treatment/Process

5) Once, a process is over, session is closed and returns to normal memory space to further execution.

This process is repeated if the process requires special treatment/secure computing

5.2.6 TRUSTED APPLICATION

According to our design, trusted application is the place, where actual business case is introduced to solve and end of the control flow from Top level (apk). For my thesis, i constructed the application to establish the secure channel communication between host and external world (for: server of the business provider). So, by replacing the code with desired business logic we can achieve the corresponding significant output.

Why secure channeling?

In real world application, trust zone works with support of both internal architecture as well as external world. Consider an example, in banking transaction, mutual authentication is required by both user and bank server in order to establish the communication between them. This is can be achieved by the challenge response. Passwords can be reused which may lead to compromise the entire communication. So challenge response is transmitting passwords change each time. Encrypting those passwords with key make the communication more secure.

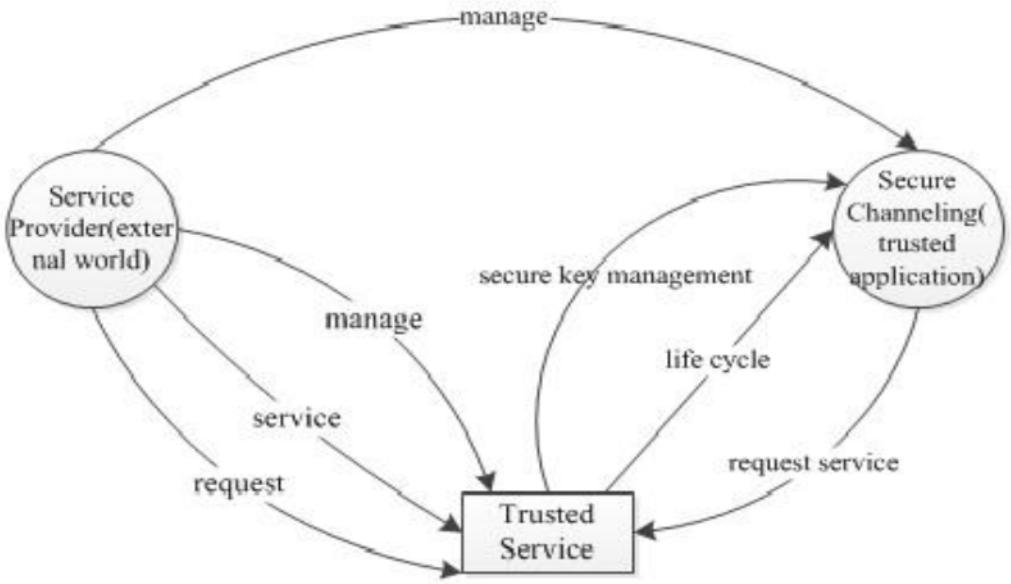


Figure 19 Trusted services

For example: Let User U wants to authenticate himself and external world system E. So U and E as agree on shared secret function F. So E sends the random message M to U and U replies with response calculated on same mechanism known by the S.

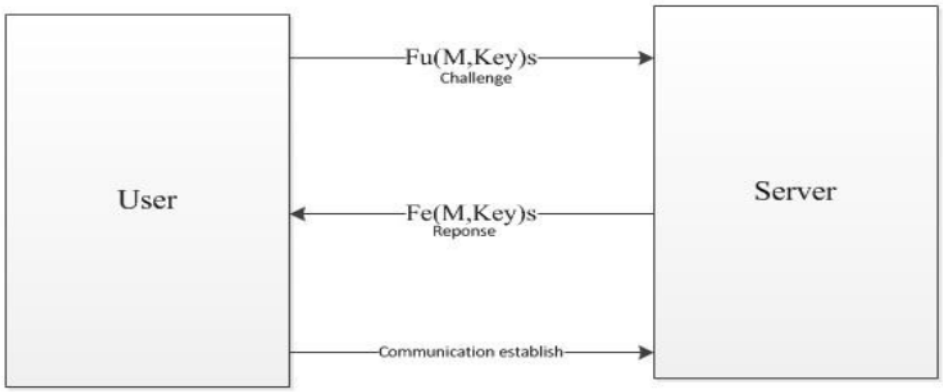


Figure 20 Communication between host and external world

In smart phone, hardware supports comes with, secure memory region to execute the mechanism (secure channeling) and application created by external service provider. So the user treats black box since he unaware about the mechanism of application. Some of the mechanisms controlled by phone manufacturers are

- Resource of hardware architecture and environment for software execution
- Installing additional application requires permits and assistance
- Billing and usage management are controlled by network operators
- Service management are controlled by service providers so subscribers are not in position to select or change the service

In our case, we use the same principles to design our trusted application. But due to these entire problems mentioned above, we stick to the basic authentication with standard encryption methods to establish the communication between the host and external world to supports our design matches with real world mechanism.

We assume in trusted application mechanism of establishing the secure channel between the mobile and external world is handled. So, trust zone handle the mechanism and input data. Secure channel means mutual authentication between the host (android emulator) and external world. This can be achieved by two way steps.

- Creation of challenge response
- Comparing cryptograms

Initially, we need to create the mutual mechanism, which can be understood by the host and external world. This mechanism is called as challenge response.

Random block of 8 bytes should be created by both the host and the external world application called as the host and external challenge.

Derivation challenge data of 16 bytes should be formed by the combining host and external challenge.

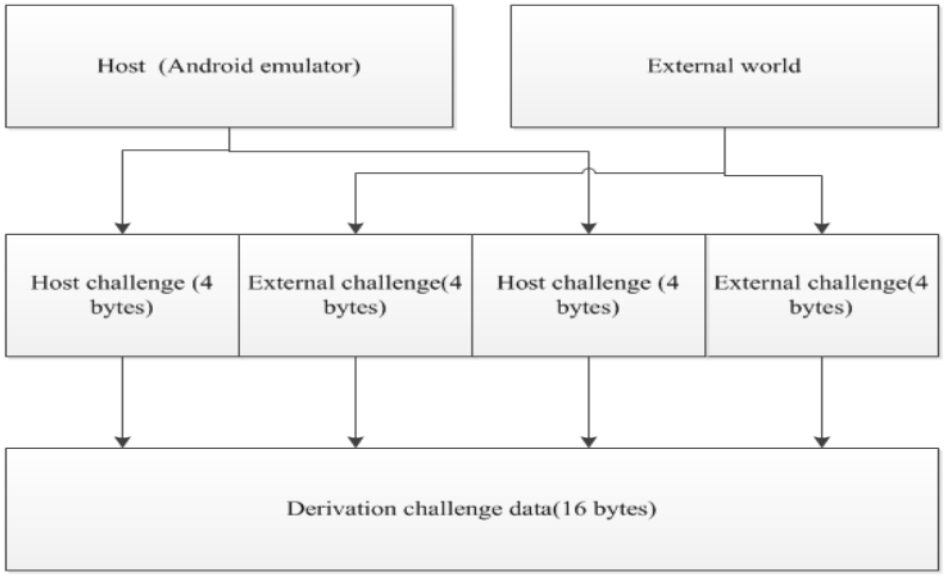


Figure 21 Creation of derivation challenge data

Encrypt the derivation challenge data with Static encryption key to form session key.

By using the session key, cryptograms are produced. Input data (credentials, pin numbers etc) are combined with the host challenge and send to external world application.

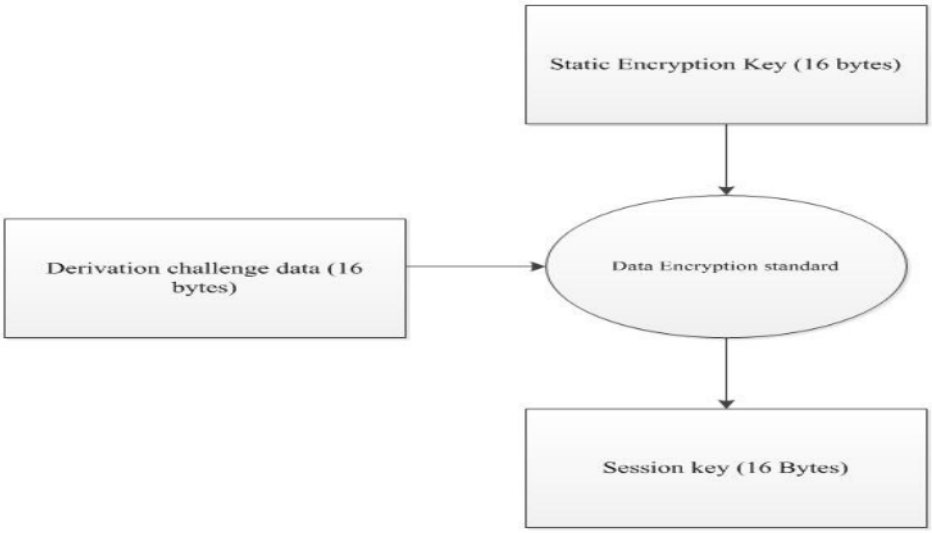


Figure 22 Creation of session key

External world application checks host cryptogram and compares it with his own cryptogram (external) generated by the same operation with data from host cryptogram. Secure channel is established if the both the operation leads the same result.

Sequence diagram: trusted application in complete control flow

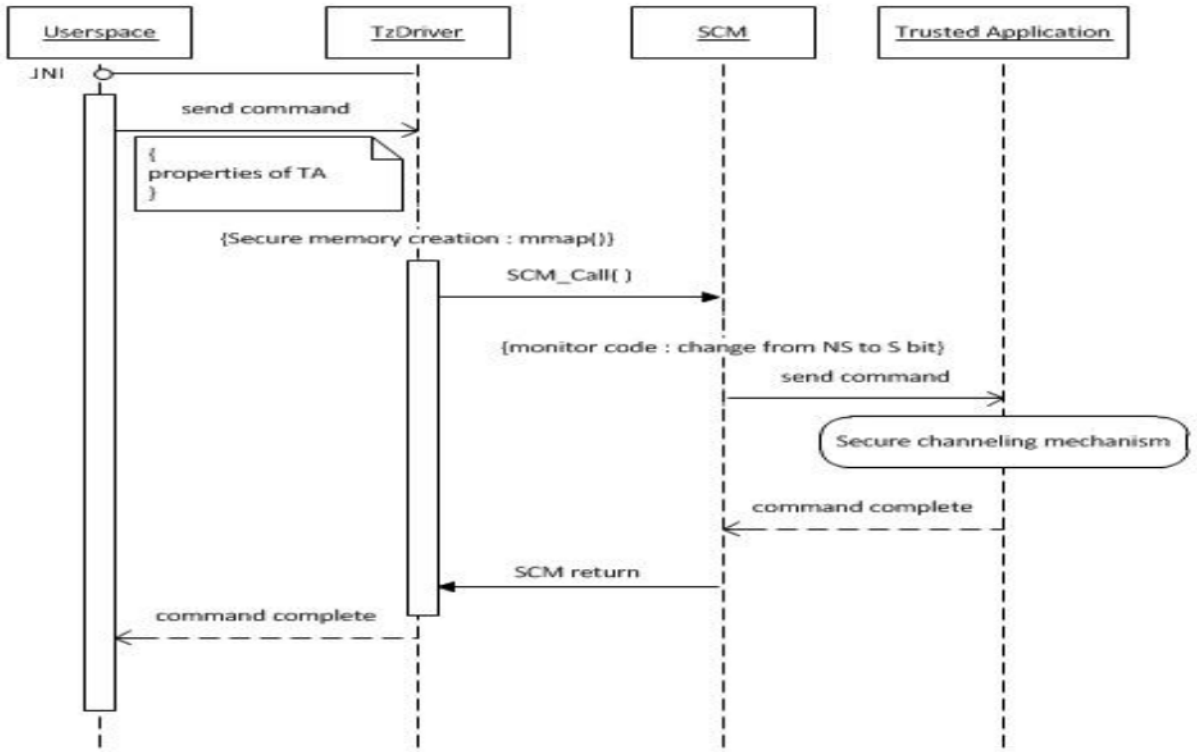


Figure 23 Sequence diagram for control flow

CHAPTER 6: EVALUATION & DISCUSSION

In this section, we will discuss about the study and implementation work carried out in this thesis process. In the beginning we discussed about the methodology support our design and then followed by, how we have achieved the trust zone in android emulator design, result and significance.

The main motto of the thesis is explained in this research question "How to extend Android emulator to support trust zone feature to develop quality operating secure system?" To provide the justification to the question, research was conducted on ARM trust zone white papers, trusted application, challenge response from various RFC's, kernel programming, system calls, android programming, JNI. So these things help me in achieving the expected design.

Design: In order to achieve the design of emulated trust zone, I have shown the evidence in analysis part and also gave various alternate designs in the same chapter with pros and cons of it.

By considering all these parameters and factors mentioned above, with all the three designs – Supervisor, dual memory and static memory design, we formulate the table to with pros and cons of each design.

	Factor	**Supervisor design**	**Dual memory design**	**Static memory design**
Parameters	**Secure memory management**	No separate memory handling for accessing two different kernel layer	Two isolated memory area to handle to two kernel region	Single strip of memory which is divided into normal and secure memory region
	Monitor mode	Supervisor will handle the control flow between two kernel region	It work same as Supervisor design, only different is supervisor need to handle control flow between two	Supervisor is just a bit variable.

			memory	
	Trusted application	It accessed first in normal kernel and then moved to secure kernel	Here, Application is accessed by normal memory first and then moved to secure memory	Trusted application is copied from normal memory to secure memory by changing the bit value of NS and S bit
	SCM_CALL	Supervisor will call for secure processing	Its work same like supervisor design	It just the procedure call to inform application require secure processing
	Interrupts	Handled by Operating system	Handled by Operating system	Handled by Operating system
	Debugging	Since all process happen in kernel space. No chance of debugging	Since all process happen in kernel space. No chance of debugging	By using NDK, we can debug the user space application but not in kernel space
	Register	Internal registers	Internal registers	Variables
Functionalities	**Complexity**	High : Since switching between kernel is not easy	High : memory handling and switching is difficult	Low: since all process happen on top level (user space) switching and handling

				memory are easy
	Reuse	No possibilities of re use of code	No possibilities of re use of code	User space application can be re used. We need to change the kernel driver according to user space program
	Implementation	Implementation is not so easy since two kernel layer involved	Implementation is not so easy since two memory layer involved	Implementation is easy
	Security	Security features are difficult to handle here	Protecting the memory is highly difficult	We can use any security standard to encrypt/decrypt the communication between host and external world

Table 5 Comparison of Designs

By seeing above all, Static memory design would more feasible compare to other two.

Result: From the analysis, we found out the actual working of ARM Trust zone as well as various ways of implementing replica of trust zone (*static memory design*) in emulator. We designed the emulated trust zone in such a way to give justification to the real working mode, even trusted application and secure channeling are also designed to provide support to the design.

Significance: By bringing Trust zone (*static memory design*) into software emulation, have plenty of significance to the outcome of the project carried in Sony Ericsson as well as great contribution to Google android development. Our Trust Zone Emulator will support the Sony

Ericsson application like SimLock, Key management, FOTA and security hardware configuration Parameter. Environment can be extended to all applications required secure treatment to process and Testing of hacking methods over the applications processed in trust zone. This project also deals with application of Trust zone emulation, cost and avoiding hardware limitation on the security of the application for open source world.

Finally, I have consolidated artifact in descending order, so that reader can easily follow the flow and idea behind each chapter and design in it.

CHAPTER 7: CONCLUSION

Our aim was to develop a software implementation of trust zone in emulator. To achieve this we analysis the existing design, understanding the feasibility of implementation of those design in emulator. After exploring three alternate design of trust zone, we come to the conclusion that, this research finds the solution to make an architecture design to implement trust zone in android emulator with merely justification to real ARM trust zone design. As the result, various architectural designs of trust zone are discussed and best one is chosen. So, the main goal for thesis is drawn.

Sub goals are also having been taken consideration in design process. Those are:

- Handling Memory segment
- Driver files
- Swapping programs to the specific memory
- Handling system call and CP15 variables
- Sample trusted application

On overall design, our framework starts with android application, navigate to JNI an interface, to provide high level platform independent experience to the developers. Based on assumption, we implement the small example called “secure channeling” that provide the construction and emulation of trusted application running in secure memory region. SCM call is made in kernel layer to communicate with external host (e.g.: server –client program) to provide mutual authentication mechanism.

For question 1:

- Alternate designs and control flow of framework will explain the overall feasibility for the construction of trust zone in android emulator. In our case, we justify that static memory design is more feasible comparatively with other designs.

For question 2:

- Since we choose static memory design (figure 14) for our case, so separate memory region (figure 16) is created for secure channeling (Appendix 2) and context switching is the navigation of control from normal memory to secure memory region and vise versa. Meanwhile other features like interrupts, application handling also done to support context switching.

We hope, our designing and research will enhance the features of smart phone application, in by future.

7.1 FUTURE WORK

The research has lot of works which provide the complete security. They are

- Implementing personalized and complied trusted application – FOTA, Banking application etc.,
- Implementing real ARM SCM call in emulator
- Designing more alternate design for trust zone
- Dependencies on hardware will be reduced
- Help future researcher to work on this
- Successor of Wallet project of Google.

BIBLIOGRAPHY

1. **David, Kleidermacher.** Bringing Security to Android based System. [Online] [Cited: April 20, 2012.] http://www.iqmagazineonline.com/current/pdf/Pg56-58_IQ_32-Bringing_Security_to_Android-based_Devices.pdf.

2. **Rao, Pradyumna Sampath Rachana.** [Online] [Cited: April 21, 2012.] http://lwn.net/images/conf/rtlws11/papers/proc/p09.pdf.

3. [Online] [Cited: March 23, 2012.] http://en.wikipedia.org/wiki/Trusted_Computing.

4. Experimental version of QEMU with basic support for ARM TrustZone (security extensions). [Online] [Cited: March 25, 2012.] https://github.com/jowinter/qemu-trustzone.

5. **Gilbert, Dr. David Alan.** Support for ARM TrustZone in QEMU. [Online] [Cited: March 25, 2012.] https://blueprints.launchpad.net/qemu-linaro/+spec/qemutrustzone.

6. **Gilje, Nils & Grimen, Harald.** *Samhällsvetenskapernas förut-sättningar.* Goteborg : s.n., 1992.

7. **ARM Security Technology.** Building a Secure System using TrustZone Technology. [Online] [Cited: March 23, 2012.] http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

8. **Anders Lindström.** *Programming of Mobile services.* Stockholm : s.n., 2011.

9. **Michael D Myers.** *Qualitative Research in Business & Management.* London : SAGE Publications Ltd , 2009.

10. **Peter M. Chisnall .** *Marketing Research.* s.l. : McGraw-Hill, 1997.

11. **Daniel Benediktsson.** *Hermeneutics: Dimensions toward LIS Thinking.* 1989.

12. **Ulf Paulsson & Maria Björklund .** *Seminar Paper: writing, presenting and opposition.* Lund : s.n., 2003.

13. **Marc Clifton& J. Dunlap.** What is SCRUM? [Online] August 2003. [Cited: May 21, 2012.] http://www.codeproject.com/Articles/4798/What-is-SCRUM#Introduction0.

14. **Lakeworks.** Scrum process.svg. [Online] January 2009. [Cited: May 15, 2012.] http://en.wikipedia.org/wiki/File:Scrum_process.svg.

15. **Leth, G. & Thurén,T.** Källkritik för Internet. [Online] 2000. [Cited: May 30, 2012.] http://www.msb.se/Upload/Produkter_tjanster/Publikationer/SPF/%C3%96vrigt/K%C3%A4llkritik%20f%C3%B6r%20Internet.pdf.

16. **Mary Poppendieck & Tom Poppendieck.** *Lean Software Development* . 2007.

17. Manifesto for Agile Software Development. [Online] 2001. [Cited: April 22, 2012.] http://agilemanifesto.org/.

18. **Winston W. Royce.** *Managing the development of large software systems.* 1970.

19. [Online] [Cited: July 2, 2012.] http://elsmar.com/Pull_Systems/img009.jpg.

20. **Patel, R& Davidsson, B.** *Att planera, genomföra ochrapportera en undersökning,.* Lund : s.n., 2003.

21. **C. Newman,A. Menon-Sen, A. Melnikov & N. Williams.** Salted Challenge Response Authentication Mechanism (SCRAM). [Online] 2010. [Cited: November 7, 2012.] http://tools.ietf.org/html/rfc5802#page-20.

22. **Donald R. Cooper & Pamela S. Schindler.** *Business research methods.* Boston : McGraw-Hill, 2003.

APPENDIX I

Coding:

Sample Code for TZ Driver: tzDriver.c

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include "myalloc.c"
#include <unistd.h>
#include "TA.h"
#include "SCM.c"
#include <sys/wait.h>
/*
Fake Monitor code ---- which intialize the TA and its properties
*/
void monitor_mode()
{
	load_TA();
	/*int i=0;
	for(i=0;i<3;i++)
	{
		printf("TA_List[%d].TA_addr: %d\n", i,
		(int)TA_list[i].TA_addr);
		printf("TA_List[%d].size: %d\n", i, (int)TA_list[i].size);
		printf("TA_List[%d].key: %s\n", i, TA_list[i].key);
		printf("TA_List[%d].id: %d\n", i, (int)TA_list[i].id);
	} */
}

int main()
{
	//char *checking_key="arun is a gud boy!";
	/* mode value before application in NSW */
	nsw_mode_value();
	monitor_mode();/* monitor and initialize the TA */
	//Basic Authentication --- from APK ID and Key is used for finding particular
	int a1 =0;
	printf("Enter the ID for TA to be loaded: ");
	scanf("%d",&a1);
	printf("output of a = %d \n ", a1);
	int i=0;
	for(i=0;i<3;i++)
	{
		if ((ID_authentication (TA_list[i].id,a1))==1)
		{
			int a =alloc_scm_command (TA_list[i].TA_addr, TA_list[i].size,
		TA_list[i].key,TA_list[i].id); /* scm_call system call */
			printf ("output from actual: %d\n", a);
		}
```

```
        }
        return 0;
}
```

Example: SCM.c

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include "myalloc.c"
#include <unistd.h>
#include "TA.h"
#include "SCM.c"
#include <sys/wait.h>

#define ram_size 128*1024*1024
#define sram_size 128*1024*1024

int scm_call (unsigned char cmd_addr)
{
        unsigned char context_id;
        unsigned char r0 = 1;
        unsigned char r1 = (unsigned char) & context_id;
        unsigned char r2 = cmd_addr;
        printf ("r0: %d\n", r0);
        printf ("r1: %d\n", r1);
        printf("r2:%d\n", r2);
        //__asm__ ("1: smc #0 @ switch to secure world\n" "cmp r0,
        #1\n" "beq 1b\n": "=r"(r0): "r"(r0), "r"(r1), "r"(r2):"r3","cc");
        return r0;
}

static int alloc_scm_command (unsigned char *addr, int size, const char *key, int id)
{
        int ret = 0;
        TA_t cmd;
        cmd. TA_addr= addr;
        cmd. Key= key;
        cmd. Size=size;
        cmd.id= id;
        unsigned char *total =(char *)malloc (ram_size+sram_size);
        unsigned char *buffer=total + ram_size;
        sw_mode_value();
        int (*pfunc)(void);
        buffer=mmap(NULL, cmd. Size, PROT_EXEC|PROT_READ|PROT_WRITE,
                MAP_PRIVATE|MAP_ANONYMOUS,1,0);
        pfunc=buffer;
        memcpy(buffer,cmd.TA_addr,cmd.size);
        printf("output of Trusted application : %d\n", pfunc());
        ret= scm_call((unsigned char )&cmd);
        printf("ret : %d\n", ret);
```

```
	return ret;
}
```

Example :TA.h

```
#include<linux/unistd.h>
#define __NR_scm_call 1079 _sys_scm_call(unsigned char *addr, int size,const char *key, int
id);
/*
Properties of Trusted application
*/

typedef struct {
	unsigned char *TA_addr;
	int size;
	// uint32_t flags;
	const char *key;
	const char *data;
	int id;
} TA_t;

typedef struct {
	int cmd;
	int status;
	int read_pos;
	int data_len;
	int data_pos;
	int data[255];
	int charactic[4];
}TA_Status;

/*struct monitor{
	int mm_value;
	int svc_value;
	int NSW_value;
	int SW_value;
}; */

void nsw_mode_value()
{
	int mm_value=0;
	int svc_value=1;
	int NSW_value=1;
	int SW_value=0;
	printf(" NSW : 0110 \n");
}

void sw_mode_value()
{
	int mm_value=1;
	int value=1;
	int NSW_value=0;
```

```
	int SW_value=1;
	printf(" SW : 1101 \n");
}

TA_t TA_list[3];

static int TA_1(void)
{
	int a=0;
	int b =9;
	int a1=0;
	int b1=9;
	//TA_2();
	return 1000;
}

static int TA_2(void)
{
	int a=02;
	int b =9;
	int a1=30;
	int b1=9;
	//TA_last();
	return (a1+b1+a+b);
}

static int TA_3(void){
	int global =1;
	return global++;
}

static int TA_last(void)
{
	return 0;
}

void load_TA()
{
	TA_list[0].TA_addr=&TA_1;
	TA_list[0].size=(int)calc_size(&TA_2,&TA_1);
	TA_list[0].key="This is a dummy key";
	TA_list[0].id=100;
	TA_list[1].TA_addr=&TA_2;
	TA_list[1].size=(int)calc_size(&TA_3,&TA_2);
	TA_list[1].key="arun is a gud boy!";
	TA_list[1].id=101;
	TA_list[2].TA_addr=&TA_3;
	TA_list[2].size=(int)calc_size(&TA_last,&TA_3);
	TA_list[2].key="Lion is the king of jungle!";
	TA_list[2].id=102;
}
```

```
int calc_size(int(TA2)(void), int(TA1)(void))
{
	return((unsigned int)TA2 - (unsigned int)TA1);
}

static int key_authentication(const char *a,const char *b)
{
	//compare(TA_list[0].key,default_key);
	if ((strcmp(a, b ))==0){
	printf("key matches \n");
	return 1;
	}
	else{
	printf("key are not matching \n");
	return 0;
	}
}

/*
Key and ID comparison for the APK and Trusted application to load corresponding Application in to secure ram
*/
int ID_authentication(int a, int b)
{
	//printf(" %d %d ", *a, b);
	int result =0;
	if (a==b){
		result=1;
		return result;
	}
	else
	{
		return result;
	}
}

unsigned int hash1(unsigned char *str,int n)
{
	printf("address: %d\n",(int)str);
	//hash = (((hash << 5) + hash) + *str) ^ hash;
	unsigned long hash = 1234;
	int i=0;
	for (i=0;i< n;i++)
	{
		//printf("str: %d\n",*str);
		hash =(*str ^ hash);
		//printf("hash: %d\n",(unsigned int)hash);
		str++;
	}
	return (unsigned int)hash;
}
```

Output

```
linux@seldlx0095:~$ gcc -o driver tzDriver.c
In file included from tzDriver.c:8:
TA.h: In function ‘load_TA’:
TA.h:92: warning: assignment from incompatible pointer type
TA.h:96: warning: assignment from incompatible pointer type
TA.h:100: warning: assignment from incompatible pointer type
TA.h: In function ‘calc_size’:
TA.h:109: warning: cast from pointer to integer of different size
TA.h:109: warning: cast from pointer to integer of different size
TA.h: In function ‘hash1’:
TA.h:147: warning: cast from pointer to integer of different size
tzDriver.c: In function ‘scm_call’:
tzDriver.c:18: warning: cast from pointer to integer of different
size
tzDriver.c: In function ‘alloc_scm_command’:
tzDriver.c:44: warning: assignment from incompatible pointer type
tzDriver.c:47: warning: cast from pointer to integer of different
size
linux@seldlx0095:~$ ./driver
NSW : 0110 /* mode change --- normal
Enter the ID for TA to be loaded : 100 / * TA ID from APK Assumed
value
output of a = 100
SW : 1101 /* mode change --- Secure
output of Trusted application : 1000 /* output of TA Application
r0: 1 // register value
r1: 127 // register value
r2:176 // register value
ret : 1 // register value
output from actual: 1 // register value
```

APPENDIX II

Memory Management code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct
{
        int isFree;
        int size;
}memBlock_t;

typedef memBlock_t *pMemBlock_t;

#ifndef TRUE
#define TRUE (1)
#endif

#ifndef FALSE
#define FALSE (0)
#endif

//heap size, modify this define if you
//want decrease or increase your heap size
#define HEAP_SIZE 1024

char *pMemStart; //Heap start address
int heapSize; //total heap size

int heapInUse; // Memory currently in use.
int blockCount; //number of block in use
char *heapEnd;  //end point of the heap
pMemBlock_t myAlloc(pMemBlock_t ,int );

enum
{
        NEW_MEMBLOCK=  0,
        NO_MEMBLOCK,
       REUSE_MEMBLOCK
};

enum
{
       FREE,
       IN_USE
};
```

```
void InitMem(char *ptr, int heapSizeInbytes)
{

        //store the ptr and heap Size In bytes in global variable
        heapSize = heapSizeInbytes;
        pMemStart = ptr;
        blockCount = 0;
        heapInUse = 0;
        heapEnd = pMemStart + heapSizeInbytes;
}

void *myalloc(int elem_size)
{
        //check whether any chunk (allocated before) is free first
        pMemBlock_t pMemBlock;
        int flag = NO_MEMBLOCK;
        pMemBlock = (pMemBlock_t)pMemStart;
        int sz = sizeof(memBlock_t);

 //check that we are not excedding heap size
 if( (elem_size + sz)  > (heapSize - (heapInUse + blockCount * sz ) ) )
 {
        printf("Max size Excedded!!!!!");
        return NULL;
 }

 while( heapEnd > ( (char *)pMemBlock + elem_size + sz)   )
 {
        if ( pMemBlock->isFree == FALSE)
        {

                if( pMemBlock->size == FALSE)
                {
                        flag = NEW_MEMBLOCK;
                        break;
                }
                if( pMemBlock->size > (elem_size + sz) )
                {
                        flag = REUSE_MEMBLOCK;
                        break;
                }
        }
        pMemBlock = (pMemBlock_t) ( (char *)pMemBlock + pMemBlock->size);
 } //end while

 if( flag != NO_MEMBLOCK)
 {
        pMemBlock->isFree = TRUE;

        if( flag == NEW_MEMBLOCK)
        {
                pMemBlock->size = elem_size + sizeof(memBlock_t);
```

```
                blockCount++;
        }
        heapInUse += elem_size;
        return ( (char *) pMemBlock + sz);
  }

  printf(" Returning as we could not allocate any memBlock_t \n");
  return NULL;
}

int MemEfficiency()
{
  /* keep track of number of blocks in a global variable */
  return blockCount;

}


void myfree(void *p)
{
        /* Mark in memBlock_t that this chunk is free */
        pMemBlock_t ptr = (pMemBlock_t)p;
        ptr--;

        blockCount--;
        ptr->isFree = FREE;
        heapInUse -= (ptr->size - sizeof(memBlock_t));
}

int main()
{
        char *pNewHeap = malloc(HEAP_SIZE);
        if(NULL == pNewHeap)
        {
                printf("System out resources!!!");
                 return -1;
         }
  memset(pNewHeap,0,HEAP_SIZE);

  InitMem(pNewHeap,HEAP_SIZE);

  char *pAllocations,*pAllocations1, *pAllocations3, *pAllocations4;

  pAllocations=myalloc(100);
  if(NULL != pAllocations)
  {
         printf("\nMemory address: %p",pAllocations);
         printf("\nBlock count: %d \t Total allocated memory from my local heap:
%d",blockCount,heapInUse);
    myfree(pAllocations);
  }
  else
```

```
  {
        printf("\nNot enough memory in heap\n");
  }

  pAllocations1=myalloc(200);
  if(NULL != pAllocations1)
  {
        printf("\n\nMemory address: %p",pAllocations1);
        printf("\nBlock count: %d \tTotal allocated memory from my local heap:
%d\n",blockCount,heapInUse);
        myfree(pAllocations1);
  }
  else
  {
         printf("Out of memory\n");
  }

  pAllocations3 = myalloc(20);
  if(NULL != pAllocations3)
  {
        printf("\n\nMemory address: %p",pAllocations3);
        printf("\nBlock count: %d \tTotal allocated memory from my local heap:
%d\n",blockCount,heapInUse);
         myfree(pAllocations3);
  }
  else
  {
         printf("\nOut of memory\n");
  }

  pAllocations4 = myalloc(300);
  if(NULL != pAllocations4)
  {
         printf("\n\nMemory address: %p",pAllocations4);
         printf("\nBlock count: %d \tTotal allocated memory from my local heap:
%d\n",blockCount,heapInUse);
        myfree(pAllocations4);
  }
  else
  {
        printf("\nOut of memory\n");
  }
}
```

APPENDIX III

Personnel Communication with Johannes Winter, for usage of SCM_Call instead of Supervisor mode in Emulated trust zone

From: Johannes Winter [johannes.winter@iaik.tugraz.at]
Sent: Monday, April 23, 2012 3:32 PM
To: Arun Muthu
Subject: help need!!! in solving the doubts in Trust zone implementation in QEMU

Hello!

On 04/20/2012 02:05 PM, Arun Muthu wrote:

Dear Winter!
>
> I am Arun Muthu, from KTH Royal institute of Technology Sweden. Right now, I am writing my master thesis in KTH under the topic adding Trust zone in qemu. My main aim is add feature of Trust zone to Qemu, and provide more power to the developer kind.With reference to github post, I came to know that, you also done the same thing (trust zone) before .I have various doubts, in developing the prototype for Trust zone in Qemu. I hope you won't mind in helping me in my thesis work.

> Some of the doubts,

> I tried to solve Trust zone in logical way - by treating Trust zone as memory unit in hosting system hardware. I am sure this might not be good idea.
> May i know, what you done as primitive or prerequisite to develop the trust zone in Qemu?

Initially we tried something similar - until we noticed that QEMU's way of simulating RAM memory is quite different to I/O peripherals. Our solution then was to adapt the ARM specific parts of QEMU's MMU simulation - effectively we treat TrustZone memory protection like an additional layer of MMU protection.

> Moreover, What SMC (Secure monitor call) instruction will do, during the execution? Flow chart or description of SMC instruction will be helpful for me in carrying out the remaining steps.
> if possible, can you send any sample code or important links to my mail id. Since i can't able to find any related post about trust zone in Github!!!

All documentation, which we used to simulate TrustZone, can be found at infocenter.arm.com - have a look at the "ARMv7-A Architecture Reference Manual".

The SMC instruction is conceptually similar to the SVC instruction, which is typcially used for system calls. You can think of SMC as kind of a "system-call form normal world kernel to secure world kernel".

Example code is a bit scarce on that topic (I am well aware of that fact). Have a look at the infocenter.arm.com website - there is a rudimentary example for TrustZone somewhere in the FAQ section. Moroever it might be interesting to consider the Fiasco L4 microkernel from TU Dresden (http://os.inf.tu-dresden.de/fiasco/). They had some TrustZone support at least in earlier versions of their micro-kernel (which can be used with our QEMU setup after a little bit of twiddling.)

Regards,
Johannes Winter
--
Johannes Winter, IAIK

Inst. for Applied Information Processing
Graz University of Technology
Inffeldgasse 16a, 8010 Graz, Austria
Phone: +43 316 873 5578
Fax: +43 316 873 5520
mailto:Johannes.Winter@iaik.tugraz.at
S E P I A @ TUG
http://www.sepia-project.eu

From: Arun Muthu
Sent: Friday, April 20, 2012 2:05 PM
To: johannes.winter@iaik.tugraz.at
Subject: help need!!! in solving the doubts in Trust zone implementation in QEMU

Dear Winter!!!,

I am Arun Muthu, from KTH Royal institute of Technology Sweden. Right now, I am writing my master thesis in KTH under the topic adding Trust zone in qemu .
My main aim is add feature of Trust zone to Qemu, and provide more power to the developer kind.

With reference to github post, I came to know that, you also done the same thing (trust zone) before.

I have various doubts, in developing the prototype for Trust zone in Qemu. I hope you won't mind in helping me in my thesis work.

Some of the doubts,

I tried to solve Trust zone in logical way - by treating Trust zone as memory unit in hosting system hardware. I am sure this might not be good idea.
May i know, what you done as primitive or prerequisite to develop the trust zone in Qemu?

Moreover, What SMC (Secure monitor call) instruction will do, during the execution? Flow chart or description of SMC instruction will be helpful for me in carrying out the remaining steps.
If possible, can you send any sample code or important links to my mail id. Since i cant able to find any related post about trust zone in Github!!!

I can be reached anytime via my cell phone, +0046-0764098299 or email id: muthu@kth.se

Your feedback would help in checking my career and thesis work.

Thanks in advance,
Yours sincerely,
Arun Muthu

******