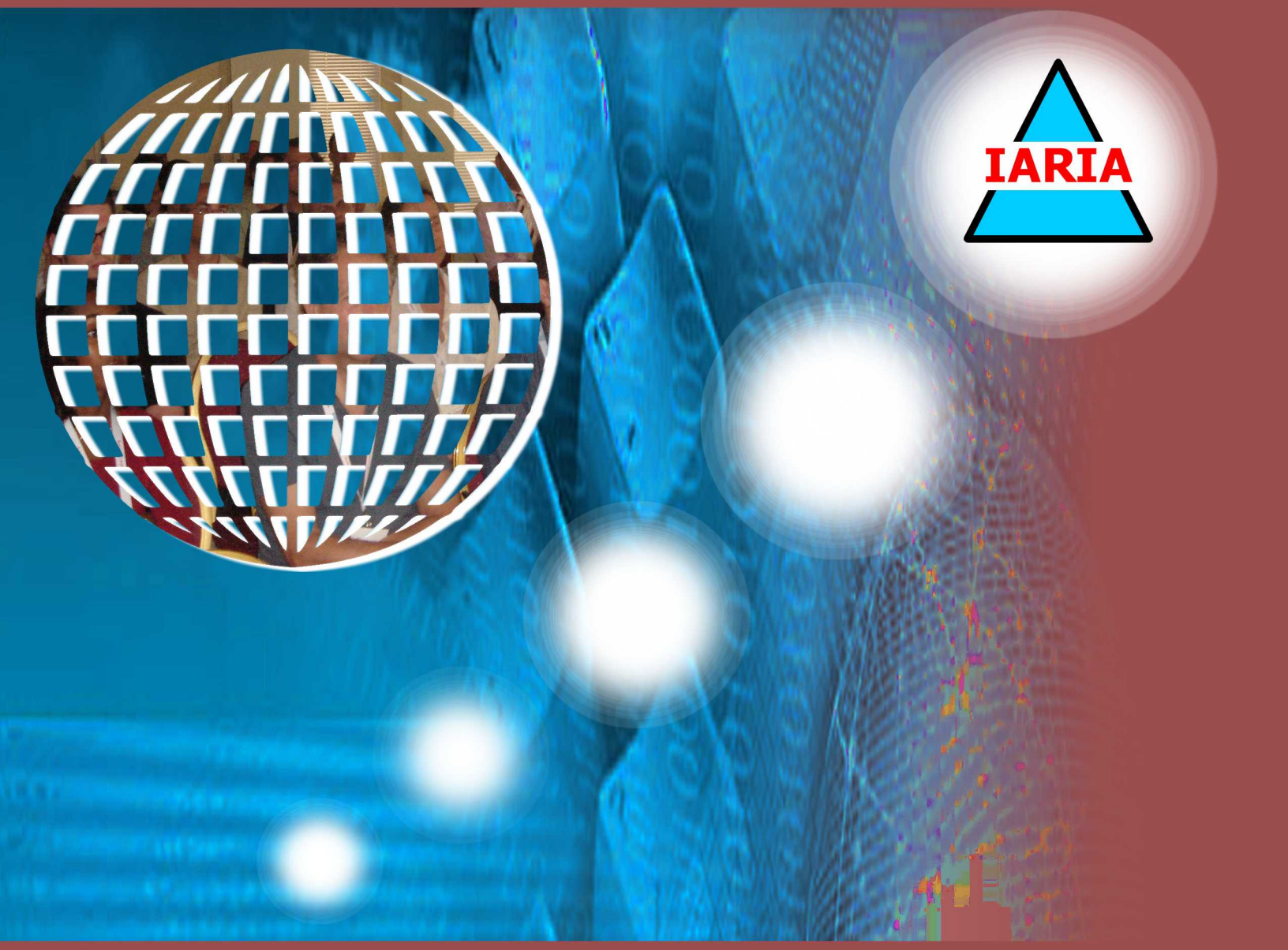
International Journal on Advances in Security



2015 vol. 8 nr. 3&4

The *International Journal on Advances in Security* is published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 8, Number 3 & 4, 2015

Raimund K. Ege, Northern Illinois University, USA
Laila El Aimani, Technicolor, Security & Content Protection Labs., Germany
El-Sayed M. El-Alfy, King Fahd University of Petroleum and Minerals, Saudi Arabia
Rainer Falk, Siemens AG - Corporate Technology, Germany
Shao-Ming Fei, Capital Normal University, Beijing, China
Eduardo B. Fernandez, Florida Atlantic University, USA
Anders Fongen, Norwegian Defense Research Establishment, Norway
Somchart Fugkeaw, Thai Digital ID Co., Ltd., Thailand
Steven Furnell, University of Plymouth, UK
Clemente Galdi, Universita' di Napoli "Federico II", Italy
Emiliano Garcia-Palacios, ECIT Institute at Queens University Belfast - Belfast, UK
Birgit Gersbeck-Schierholz, Leibniz Universität Hannover, Germany
Manuel Gil Pérez, University of Murcia, Spain
Karl M. Goeschka, Vienna University of Technology, Austria
Stefanos Gritzalis, University of the Aegean, Greece
Michael Grottke, University of Erlangen-Nuremberg, Germany
Ehud Gudes, Ben-Gurion University - Beer-Sheva, Israel
Indira R. Guzman, Trident University International, USA
Huong Ha, University of Newcastle, Singapore
Petr Hanáček, Brno University of Technology, Czech Republic
Gerhard Hancke, Royal Holloway / University of London, UK
Sami Harari, Institut des Sciences de l'Ingénieur de Toulon et du Var / Université du Sud Toulon Var, France
Dan Harkins, Aruba Networks, Inc., USA
Ragib Hasan, University of Alabama at Birmingham, USA
Masahito Hayashi, Nagoya University, Japan
Michael Hobbs, Deakin University, Australia
Neminath Hubballi, Infosys Labs Bangalore, India
Mariusz Jakubowski, Microsoft Research, USA
Ángel Jesús Varela Vaca, University of Seville, Spain
Ravi Jhawar, Università degli Studi di Milano, Italy
Dan Jiang, Philips Research Asia Shanghai, China
Georgios Kambourakis, University of the Aegean, Greece
Florian Kammueller, Middlesex University - London, UK
Sokratis K. Katsikas, University of Piraeus, Greece
Seah Boon Keong, MIMOS Berhad, Malaysia
Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark
Marc-Olivier Killijian, LAAS-CNRS, France
Hyunsung Kim, Kyungil University, Korea
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong

Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy
Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan
Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Vladimir Privman, Clarkson University, USA
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, Institute for Infocomm Research (Member of A*STAR), Singapore
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Vladimir Stantchev, Institute of Information Systems, SRH University Berlin, Germany
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland
Enrico Thomae, Ruhr-University Bochum, Germany

International Journal on Advances in Security
Volume 8, Numbers 3 & 4, 2015

CONTENTS

pages: 109 - 119
Terminal Virtualization Framework for Mobile Services
Tao Zheng, Orange Labs International Center Beijing, China
Song Dong, Orange Labs International Center Beijing, China

pages: 120 - 129
Chronomorphic Programs: Runtime Diversity Prevents Exploits and Reconnaissance
Scott Friedman, SIFT, LLC, USA
David Musliner, SIFT, LLC, USA
Peter Keller, SIFT, LLC, USA

pages: 130 - 140
Building Trusted and Real Time ARM Guests Execution Environments for Mixed Criticality With T-KVM, a hypervisor architecture that implements an hardware isolated secure and real time environment
Michele Paolino, Virtual Open Systems, France
Kevin Chappuis, Virtual Open Systems, France
Alvise Rigo, Virtual Open Systems, France
Alexander Spyridakis, Virtual Open Systems, France
Jérémy Fanguède, Virtual Open Systems, France
Petar Lalov, Virtual Open Systems, France
Daniel Raho, Virtual Open Systems, France

pages: 141 - 152
The Dichotomy of Decision Sciences in Information Assurance, Privacy, and Security Applications in Law and Joint Ventures
Simon Reay Atkinson, Centre for International Security Studies, FASS, University of Sydney., Australia
Gregory Tolhurst, Faculty of Law, University of Sydney., Australia
Liaquat Hossain, Information Management Division of Information and Technology Studies, The University of Hong Kong, Hong Kong

Terminal Virtualization Framework for Mobile Services

Tao Zheng, Song Dong
Orange Labs International Center
Beijing, China
e-mail: {tao.zheng | song.dong}@orange.com

***Abstract* - Terminal virtualization focuses on applying Information Technology (IT) virtualization technology to the terminals, and realizes the full or parts of terminal functions extension or migration to other devices on the network, such as resource reducing, information sharing, data synchronization, etc. It is becoming increasingly clear that more and more features of terminal virtualization and mobile computing on the edge will be used in practice. However, some issues are raised with terminal virtualization, such as security, privacy, Quality of Service (QoS), efficient transmission, computation/functions offloading management, etc. In this paper, after analyzing above issues, a mobile terminal virtualization framework is proposed to solve them. Then the implementation methods of the proposed framework modules are presented in detail, which are based on popular the terminal Operating System (OS) and some current technologies. This framework provides a better transparent experience to users from free handover on network to unified sensors usage.**

Keywords - terminal virtualization; mobile cloud computing; computation offloading; QoS; Content Centric Networking (CCN); fountain code; Quality of Experience (QoE).

I. INTRODUCTION

This paper extends our earlier work [1] presented at the Eleventh International Conference on Networking and Services (ICNS 2015).

With the explosive growth of mobile terminals in recent years, user preferences have shifted from traditional cell phones and laptops to smartphones and tablets. In recent years, there are abundant applications in various categories, such as entertainment, health, games, business, social networking, travel and news, running at mobile terminals. The burden of computation on the terminals has been raised rapidly and more functions and sensors are required to be applied to them. Mobile cloud computing and terminal virtualization are proposed to handle these issues, which are able to provide tools to the user when and where it is needed irrespective of user movement, hence supporting location independence. Indeed, "mobility" is one of the characteristics of a pervasive computing environment where the user is able to continue ones work seamlessly regardless of the movement.

Advances in the portability and capability of mobile terminals, together with widespread Long Term Evolution (LTE) networks and WiFi access, have brought rich mobile application experiences to end users. Undoubtedly, mobile broadband terminals, such as smart phones, tablets, wireless dongles and some data-intensive apps have caused an exponential increase in mobile Internet Protocol (IP) data usage, and they use up the mobile bandwidth. The demand for ubiquitous access to a wealth of media content and services will continue to increase, as indicated in a report by Cisco [2]: the Compound Average Growth Rate (CAGR) of global IP traffic from mobile terminals is 57% from 2014 to 2019, which is triple CAGR from fixed Internet.

In addition, the resource-constrained mobile terminals, especially with limited battery life, have been a barrier to the improvements of mobile applications and services. While new smart phones with bigger screens, faster Central Processing Units (CPUs), and larger storage are launched continually, and the bandwidth of wireless networks has increased hundreds of times from the second-generation wireless telephone technology to the fourth-generation wireless telephone technology in just a few years, the development of batteries has lagged far behind the development of other components in mobile terminals. In fact, faster CPUs, larger displays and multimedia applications consume more battery energy. The limitations of computation resources and sensors are other stumbling blocks for services development. Mobile cloud computing and terminal virtualization can help to resolve this issue.

Mobile cloud computing and terminal virtualization have been the leading technology trends in recent years. The increasing usage of mobile computing is evident in the study by Juniper Research, which states that the consumer and enterprise market for cloud-based mobile applications is expected to rise to $9.5 billion by 2014 [3]. Mobile cloud computing/terminal virtualization is introduced to resolve the conflicts mentioned above, where the cloud serves as a powerful complement to resource-constrained mobile terminals. Rather than executing all computational and data operations locally, mobile cloud computing/terminal virtualization takes advantage of the abundant resources in cloud platforms to gather, store, and process data for mobile terminals. Many popular mobile applications have actually employed cloud computing to provide enhanced services. More innovative cloud-based mobile applications like healthcare monitoring and multiplayer online mobile games are also under development.

The objective of the paper is to introduce the concept of terminal virtualization and study the related issues and research status. On this basis, a proposal terminal virtualization framework is finally presented and discussed on the implementation. This framework provides a better transparent experience to users through utilizing various techniques and based on the current terminal OS.

This paper is organized as follows. In Section II, we introduce the concept and current status of terminal virtualization. In Section III, capabilities and functions extension are analyzed. In Section IV, a terminal virtualization framework for mobile networks is presented. Implementation method of the proposed framework in detail is presented in Section V. Finally, Section VI summarizes the conclusions.

II. TERMINAL VIRTUALIZATION

Terminal virtualization helps to relieve the local resource-constrained problem through offloading some tasks to the cloud and utilizing capabilities and functions in the cloud. First, the scope of terminal virtualization needs to be clarified. Then, drivers and benefits are proposed. Finally, some challenges are raised by terminal virtualization.

A. The scope of terminal virtualization

From the virtualization point of view, mobile cloud computing can support a part of terminal virtualization scenario. There are two scenarios as following.

- Full Virtualization Scenario

The requirement for full terminal virtualization mainly comes from some enterprises. In these enterprises, employees are buying their own terminals and want to connect to the enterprise network so that they can do their work with greater flexibility. However, the employees also do not want to give up user experience and freedom at the cost of complex IT security policies. In order to achieve this goal, terminal virtualization is becoming a very attractive choice because it offers flexibility and addresses the concerns over privacy of personal data while also delivering the security requirements of the enterprise. On the other side of the ecosystem, the terminal makers and carriers will benefit from terminal virtualization because they are able to more easily replicate the features found in various terminals and also deliver more features at a lower cost.

Full terminal virtualization is not an ordinary schema for public mobile customers. In general way, the terminal is sold with a pre-determined OS and customers can use services based on this OS.

- Partial Virtualization Scenario

Broadly speaking, mobile cloud computing can be as one kind of partial terminal virtualization, a part of terminal computation powers and functions can be virtualized into the remote networked cloud. Terminals can get local experience through running remote apps or some information located in the remote cloud.

This scenario is more practiced and popular at present. Some applications employ this method to add enhanced functions or improve user experience. Even cloud phone appears and is deeply merged with networking services for user convenience. In this paper, we mainly focus on the partial virtualization scenario.

B. Drivers and benefits of terminal virtualization

Terminal virtualization facilitated the fusion of mobile terminal and cloud service that provides a platform wherein some computing, storing and data abstraction tasks are performed by the cloud and mobile terminal simply seeks an access to them. In the following, we show the drivers and benefits of terminal virtualization.

- Limitless Storage Space

Now, instead of memory cards for more space, the cloud storage can provide limitless space for applications, even with the help of terminal virtualization framework/middleware they do not need to care about the location of the storage.

- Improved Processing Facility

The price of a mobile terminal is largely dependent on its CPU's speed and performance. With the help of terminal virtualization, all the extensive and complex processing is done at the cloud level, thereby enhancing the mobile terminal's performance without upgrading the terminal's CPU.

- Save Radio Access Network (RAN)/Access Bandwidth & Resources

With the tremendous increase in mobile bandwidth consumers and user's throughput, RAN/access resources have become more valuable than before. When some functions and computation tasks are offloaded into cloud, the result instead of the original metrical is sent to the terminal, so the RAN/access bandwidth can be saved for other use.

- Enhanced Battery Life

Terminal virtualization lends a very strong helping hand to battery life of terminals. With most of the processing handled by the cloud, the battery life is enhanced, thereby making the most optimum use of the remaining recharge cycles.

- Improved User Experience

The above mentioned features will improve the end-user experience substantially, especially the experience from low-end terminals.

- Economic Factors

For the consumers, terminal virtualization can bring some new functions and improved capabilities to the old terminal without spending one penny. For operators, the benefits come from saved network resources and flexible service deployment by terminal virtualization.

- Reserving for Upcoming Technologies

Terminal virtualization is adapted to the tremendous pace of development technologies and works most efficiently with the upgrades. Through separating the implementation from the function body, upcoming technologies can be easily introduced to the terminals.

C. Challenges

In this section, we argue that some issues in terminal virtualization have not been sufficiently solved satisfactorily.

- Energy-efficient Transmission

Wireless networks are stochastic in nature: not only the availability and network capacity of access points vary from place to place, but the downlink and uplink bandwidth also fluctuates due to weather, building/geographical shields, terminal mobility, and so on. Measurement studies [4] show that the energy consumption for transmitting a fixed amount of data is inversely proportional to the available bandwidth.

Computation/Data offloading can save energy only if heavy computation is needed and a relatively small amount of data has to be transferred. Energy efficiency can be substantially improved if the cloud stores the data required for computation, reducing data transmission overhead. Bandwidth allocation and admission control mechanisms in cellular base stations and access points may guarantee network connectivity to a certain extent, but cannot eliminate the stochastic nature of wireless links. An alternative approach is to dynamically adjust application partitioning between the cloud and mobile terminals according to network conditions, although it is challenging to quickly and accurately estimate the network connectivity with low overhead.

Energy-efficient transmission is also critical when exploiting the cloud to extend the capabilities of mobile terminals. Frequent transmissions in bad connectivity will overly consume energy, making the extended capabilities unattractive, as battery life is always the top concern of mobile users. A solution called eTime [5] is to adaptively seize the timing opportunity when network connectivity is good to pre-fetch frequently used data while deferring delay-tolerant data.

- Security

There are several aspects of terminal virtualization security, including antivirus, authentication, data protection, and digital rights management. Security vulnerability can cause serious problems, including property damage, cloud vendor economic loss, and user distrust. Since mobile terminals are resource-constrained, locally executed antivirus software can hardly protect them from threats efficiently. A current solution is to offload the threat detection functionality to the cloud. Nevertheless, since a pure cloud antivirus relies on cloud resources, it is difficult to deal with malware that can block the terminal's Internet connection.

Besides, authentication is critical for access to sensitive information, such as bank accounts and confidential files. With constrained text input on mobile terminals, users tend to use simple passwords, making mobile applications more vulnerable to authentication threats. To solve this issue, Chow et al. [6] build up an authorization platform where users are identified by their habits (e.g., calling patterns, location information, and web access). The platform routinely records user behavior information. When a server receives an authorization request, it redirects the request to an authorization engine, which uses the aggregated behavior information and an authorization policy to decide whether to accept the request or not.

- Privacy

Since mobile terminals are usually personal items, privacy must be considered when leveraging the cloud to store and process their confidential data remotely.

A secure data processing framework [7] can be used into terminal virtualization, where critical data are protected by the unique encryption key generated from the user's trusted authority and stored in an area named Extended Semi-Shadow Image isolated from the public domain. Even when storage is breached in the cloud, unauthorized parties including the cloud vendor cannot obtain the private data.

Another particular privacy issue for mobile users is the leakage of personal location information in location-based services. To address the issue, a method called "location cloaking" [8] makes user location data slightly imprecise before submitting them to the cloud. But the imprecise data sometimes cannot provide relevant or satisfactory results to users in certain applications. Therefore, location cloaking should be adaptively tuned to balance the trade-off between privacy and result accuracy.

- Real-time Requirements and Service QoS

When terminal virtualization and mobile computing are applied, QoS will become more important. How to guarantee the related data or stream to be transmitted in time determinates the services' failure or success.

While different applications offer different functionality to end users, the primary service Key Quality Indicators (KQIs) across the application's customer facing service boundary for end users of applications generally include service availability, service latency, service reliability, service accessibility, service throughput, and application specific service quality measurements.

III. COMPUTATION OFFLOADING & FUNCTIONS EXTENSION

Terminal virtualization enables enhanced mobile experiences that were previously impossible on resource-constrained and function-constrained mobile terminals. Many commercial mobile applications use the cloud to bring about rich features. They usually employ a client-server framework that consists of two parts, which run on the mobile terminal and the cloud, respectively. Essentially, cloud computing helps extend the capabilities and functions of mobile terminals in some aspects.

A. Capabilities & Functions extension

Through terminal virtualization, the capabilities and functions can be reallocated between terminal and cloud, as shown in the following examples

- Computation-intensive Task

At present, many applications nowadays support speech/picture/video recognition. The models for recognition and high-quality synthesis must be trained with millions of samples in thousands of examples. This computation-intensive task is infeasible on a mobile terminal and should be offloaded to the cloud.

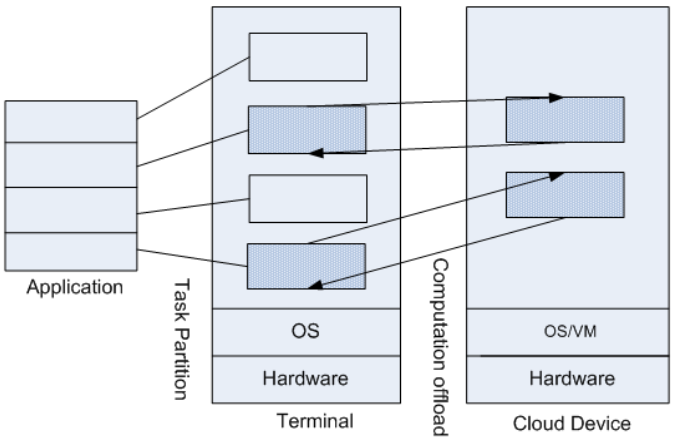


Figure 1. The procedure of computation offloading

- Remote Sensors/Inductors

Because of the limitation from terminal itself (low-end model lacking some sensors or inductors) or other conditions (e.g., distance exceeding the maximum length of sensors), some services cannot work well. However, these services can work through getting and storing related information from mobile cloud platform. From the terminal point of view, its functions are extended.

- Application Portability

It allows for the rapid transfer of applications (which may occur on the fly), providing the freedom to optimize, without the constraints of the location and required resources of the virtual appliances. The precise but extensible definition of the services provided by the application platform is the key to ensuring application portability.

B. Computation offloading Decision

To overcome resource constraints on mobile terminals, a general idea is to offload parts of resource-intensive tasks to the cloud (centralized server or other peers). Since execution in the cloud is considerably faster than that on mobile terminals, it is worth shipping code and data to the cloud and back to prolong the battery life and speed up the application. This offloading procedure is illustrated in Fig. 1. The application will be separated to several computation units/tasks according to functions or calling relations. Then some of the computation units/tasks will be offloaded to could devices to be executed remotely. At last, the executing results will be returned to the terminal as if the total application is executed locally.

There are several technologies to realize the runtime environment in the cloud, and the major differences between offloading techniques lie in the offloading unit and partitioning strategies.

- Client–Server Communication Mechanism

In the Client–Server Communication, process communication is done across the mobile terminal and cloud server via protocols, such as Remote Procedure Calls (RPC), Remote Method Invocation (RMI) and Sockets. Both RPC and RMI have well supported APIs and are considered stable by developers. However, offloading through these two methods means that services need to have been pre-installed in the participating terminals.

Spectra [9] and Chroma [10] are the examples of systems that use pre-installed services reachable via RPC to offload computation. Hyrax [11] has been presented for Android smartphone applications, which are distributed both in terms of data and computation based on Hadoop ported to the Android platform. Another framework based on Hadoop is presented in [12], for a virtual mobile cloud focusing on common goals where mobile terminal are considered as resource providers. Cuckoo [13] presents a system to offload mobile terminal applications onto a cloud using a Java stub/proxy model. The Mobile Message Passing Interface (MMPI) framework [14] is a mobile version of the standard Message Passing Interface (MPI) over Bluetooth where mobile terminals function as fellow resource providers.

- Mobile Agent

Scavenger [15] is another framework that employs cyber-foraging using WiFi for connectivity, and uses a mobile code approach to partition and distribute jobs. Using its framework, it is possible for a mobile terminal to offload to one or more agents and its tests show that running the application on multiples in parallel is more efficient in terms of performance. However, the fault tolerance mechanism is not discussed and since its method is strictly about offloading on agents and not sharing, it is not really dynamic. Also, its agents are all desktops and it is unclear if Scavenger is too heavy to run on mobile phones.

- Virtualization/Virtual Machine (VM) Migration

The execution cannot be stopped when transferring the memory image of a VM from a source terminal to the destination server [16]. In such a live migration, the memory pages of the VM are pre-copied without interrupting the OS or any of its applications, thereby providing a seamless migration. However, VM migration is somewhat time-consuming and the workload could prove to be heavy for mobile terminals.

VM migration is used by a majority of frameworks, including Cloudlets [17], Maui [18], CloneCloud [19], and MobiCloud [20]. Virtualization greatly reduces the burden on the programmer, since very little or no rewriting of applications is required. However, full virtualization with automatic partitioning is unlikely to produce the same fine grained optimizations as that of hand coded applications, although rewriting each and every application for code offload is also not practical. Maui actually does not rely on pure VM migration as done in CloneCloud and Cloudlets, but uses a combination of VM migration and programmatic partitioning. However, in cases where the mobile terminal user is within range of an agent terminal for a few minutes, using VM migration may prove to be too heavyweight, as is pointed out by Kristensen [15], which uses mobile agents in light of its suitability in a dynamic mobile environment.

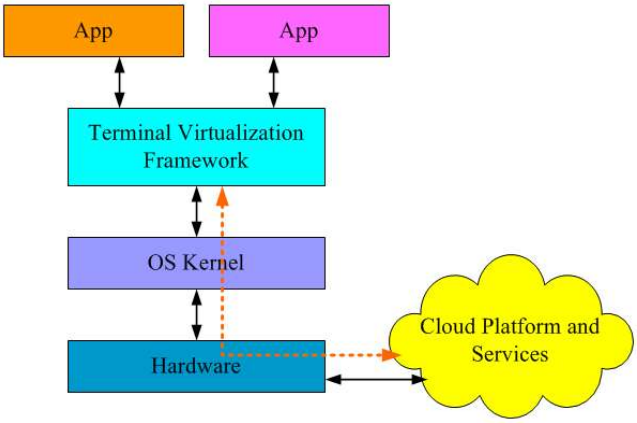


Figure 2. Hierarchical structure of the framework

C. Applications

The following lists the current applications using terminal virtualization concept, from functions extension to complex computing tasks.

- Mobile Cloud Phone

Mobile cloud phone differs from other smart phones in that it does not need to download and store apps and content on the phone; it instead accesses personal information and runs programs stored on remote network servers, via the cloud.

YunOS 3.0 [21], developed by Alibaba, debuts officially with cloud-based service for movie, taxi and other reservations on October 20th, 2014. It comes with the brand new service Cloud Card, which runs entirely in the cloud and offers the user the option to select movie tickets, taxi services and more.

- Cloud Storage and Video Adaption

Through terminal virtualization, some part of data that is stored in the cloud instead of stored on the terminal can be treated as local data. And video stored in cloud platform can be adapted to appropriate format and code streaming fitting for the terminal when the terminal requests this video.

- Image and Natural Language Processing

For this kind of applications, the complex computation jobs, which are difficult for local operating OS, should be offloaded to the cloud platform and the mobile terminal just holds some interface functions. Image and voice recognition, search, adaption, natural language translation, and Artificial Intelligence (AI) machine conversation, etc., which belong to this kind of applications, can be implemented on some low-end phones with the help of computation offloading.

- Augmented Reality (AR)

Algorithms in augmented reality are mostly resource and computation-intensive, posing challenges to resource-poor mobile devices. These applications can integrate the power of the cloud to handle complex processing of augmented reality tasks. Specifically, data streams of the sensors on a mobile device can be directed to the cloud for processing, and the processed data streams are then redirected back to the device. It should be noted that AR applications demand low latency to provide a life-like experience.

IV. Proposed Framework for Mobile Services

This section proposes a mobile terminal virtualization framework based on mobile OS. The framework locates in the middleware layer between OS kernel and applications, as shown in Fig. 2, which proxies the calls between the apps and the OS and provides the virtualization function to apps using networks, remote cloud platforms and services through OS, hardware and network.

A. The framework overview

The framework illustrated in Fig. 3 is composed of four processing modules: application virtualization module, computation virtualization module, storage virtualization module and network virtualization module, and a management module.

In the framework, processing modules are in charge of receiving and responding the callings from applications to OS. Management module is used to manage the framework, including security management, configuration management, network and cloud service monitoring, etc.

B. Component Modules

As shown in Fig. 3, the processing modules are able to choose the best method to process the calling according to the current status of mobile network bandwidth, local resources, terminals hardware limitation and remote cloud resources. Meanwhile, the framework provides local calling responses to the applications and shields the actual calling responses.

- Application Virtualization Module

This module is in charge of processing the functions extension of applications. When the application accesses the terminal's hardware, for example one kind of sensor, this module will check if it is available. If not, this module is responsible for finding a same remote available sensor in the cloud to satisfy the application's demand and providing the response with the result from the remote sensor to the application.

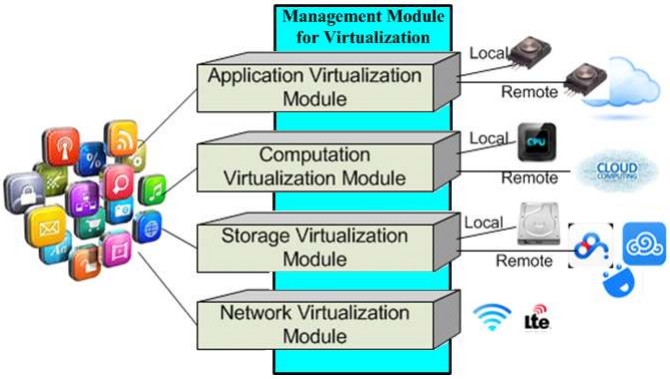


Figure 3. The functions of processing modules

- Computation Virtualization Module

This module takes charge of monitoring the terminal's computation resource and analyzing the computation request from applications. When the computation resource is constrained (for example, the CPU usage is more than 85%) or some applications with sophisticated computing power are run (for example, virtual reality service, language processing, etc., it can be configured in advance), this module will offload some computation tasks to the remote cloud server and provide the processing result to the applications.

- Storage Virtualization Module

This module is in charge of monitoring the terminal's storage resource and providing the remote cloud storage to applications. Through this module, the local applications can use cloud storage provided by different Service Providers (SPs), such as Baidu, Tencent, Huawei, etc., as using a local storage. At the same time, this module monitors the speed and status of the remote cloud storages and provides the best one to applications.

- Network Virtualization Module

This module is in charge of monitoring the terminal's network status and providing the best one or binding different network accesses to increase the data throughput according to the applications' demand.

- Management Module

The management module is responsible for managing the framework. The security function including network security and resources security is an important function in this module. Other management functions include all kinds of configuration management, for example, some resources and offloading thresholds, and remote resource monitoring, for example, all kinds of cloud services, network status.

V. FRAMEWORK IMPLEMENTATION

We are implementing an early-phase prototype based on Android OS according to the proposed framework in our lab. The following part discusses the implementation of the modules, where network virtualization module and the storage virtualization module are relatively easier to be implemented than other three modules in the framework.

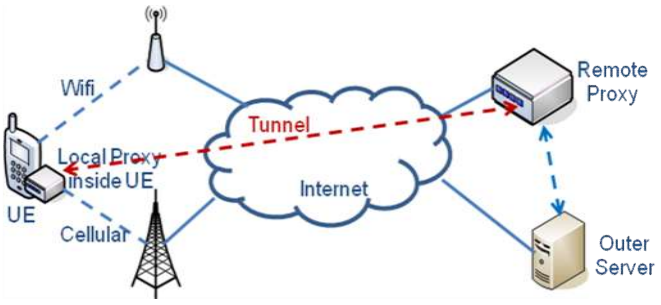


Figure 4. The overview of network virtualization function

A. Network Virtualization Implementation

Network virtualization means that services and applications just make the networking call and do not have to be concerned with the details of network environment, e.g., the network type (WiFi/3G/4G), the network failure and recovery, how to use multiple network paths (priority/traffic offloading) .

The network virtualization module employed a method [22] to implement the network access independence, for example, using multiple access paths simultaneously, switching between access paths according to the current network environment, and recovering the access path automatically. A local proxy in terminal and a remote proxy in network cooperate to implement the functions of network virtualization module. And the applications and services are able to automatically adapt the change of network and are not affected by it.

The system overview is illustrated in Fig. 4. It is composed of terminal local proxy and remote proxy. A local proxy built in the terminal is responsible for relaying the conversation between applications and network. The remote proxy exchanges Data packets with the local proxy through the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) based tunnel established over multiple heterogeneous links in the content centric way.

The remote proxy fetches the content from the outer server in the Internet to satisfy the terminal's request. The main reason for using two proxies is that the tunnel between the two proxies can help Content Centric data stream penetrate the network.

The working environment can be IPv4 or IPv6. In IPv6 network, the remote proxy can be assigned with different IPv6 addresses including unicast address, multicast address and anycast address. Given the unicast address, the terminal establishes the tunnel with the single specific remote proxy. The multicast addressing refers to the configuration where the terminal can set up the tunnels concurrently connecting the collection of remote proxies. The anycast addressing makes the terminal build the tunnel to the nearest remote proxy among several candidates.

Since the service session in the scheme is identified with the Uniform Resource Identifier (URI) that is independent of the IP addresses associated with the different connections to the network, the terminal can keep the ongoing session alive as long as the content identifier is invariant. The dynamics due to the connection switching in the mobile scenario is only visible in the tunnel running over the TCP or UDP sessions managed by the local proxy.

CCN is an alternative approach to the architecture of networks, which was proposed by Xerox PARC within the Content Centric Networking (CCNx) [23] project. From the network perspective, in CCN, network entities in ordinary network are replayed by data entities.

Unlike state-of-the-art multi-path approaches such as Multi-Path Transmission Control Protocol (MP-TCP), CCN can help us to hide some network control implementing details with the help of the 'connection-less' nature of CCN. In the network virtualization solution, we utilize the CCN concept rather than

CCN protocol, thus it is not necessary to consider the change of network access paths through fountain codes to encapsulate data packets.

The service session coupling with the remote proxy avoids the problem of Domain Name System (DNS) resolution potentially confronted with the multi-homed terminal. Since the remote proxy acts as the agent of the terminal for content acquisition, the terminal has no need for DNS resolution except to forward the request message to the remote proxy. The update on the access router is additionally not mandatory anymore because the conversion between IP address and URI is executed in the sense of application layer.

In this solution, fountain codes are used to transport data packets between two proxies. The reasons of choosing fountain codes as the encapsulation technology are shown as follows. Fountain codes are rateless in the sense that the number of encoded packets that can be generated from the source message is potentially limitless. Regardless of the statistics of the erasure events on the channel, the source data can be decoded from any set of K' encoded packets, for K' slightly larger than K. Fountain codes can also have very small encoding and decoding complexities and automatically adapt the change of multiple access paths to avoid the implementation of path control details [24].

In this module, the source hosts simply transfer as many packets with the different coding schemes as possible to the destination hosts without concerns over the reordering induced in various paths. It increases the data throughput and avoids the complicated reconciliation between the multiple paths.

The local proxy and remote proxy, fountain encoding and decoding had been implemented in our lab. The test-bed is illustrated in Fig. 5. Because there is no mobile data access in the lab, we chose two WiFi interfaces and Access Points (AP) to simulate two access paths. On this test-bed, we tested various combinations of two access paths and handover between them. The data delivery between the laptop and Internet cannot be interrupted during the transition among these scenarios. The test result validated the feasibility and validity of network virtualization solution.

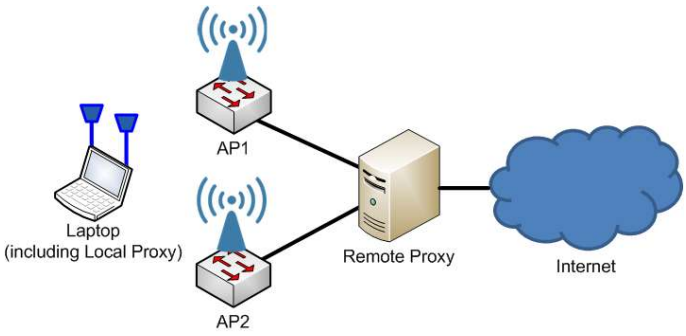


Figure 5. The test-bed for network virtualization

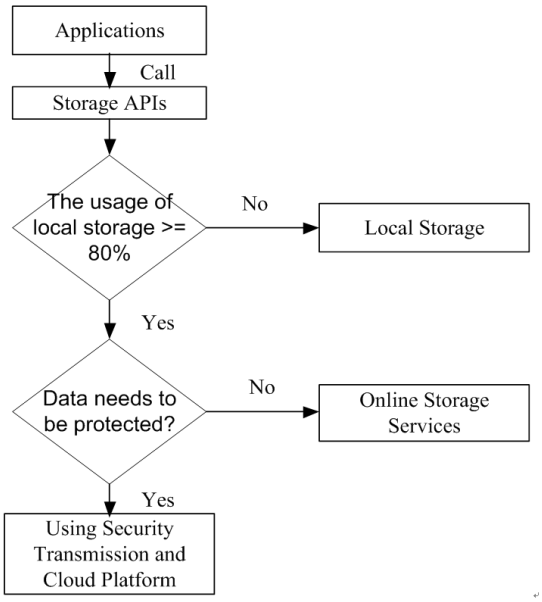


Figure 6. The flowchart of storage virtualization function

B. *Storage Virtualization Implementation*

The storage virtualization module adds online storage services/cloud platforms into the system storage as a directory, which can be accessed by the applications like a local one. When the directory is accessed, the storage virtualization module will automatically exchange the data with the online storages/cloud platform. Another issue we need to be considered is security, including the storage location security and the transport security. The detail implementation of security is discussed in part E in Section V.

The flowchart of storage virtualization function is shown in Fig. 6.

When the system storage APIs are called, the usage of local storage will determine whether to call the remote storage APIs. If the remote storage is called, the confidentiality data will be stored in the remote safe platform through secure transmission mode (refer to part E) and the data without security requirement will be transported to some online storages through Internet, such as Baidu, Huawei online storage services.

C. *Application Virtualization Implementation*

To implement application virtualization, some operating system calls accessing local hardware need to be intercepted and rewritten. The functions to access online hardware resource will be added into the application virtualization module.

For example, some kinds of sensors are supported by Android OS, which are defined in APIs. The sensors include accelerometer sensor, gravity sensor, gyroscope sensor, light sensor, linear acceleration sensor, magnetic field sensor, proximity sensor, rotation vector sensor, temperature sensor and pressure sensor. The last two sensors are relatively rare in most of the terminals. So, the remote sensors can be used to get some information for the applications as local calls through the application module.

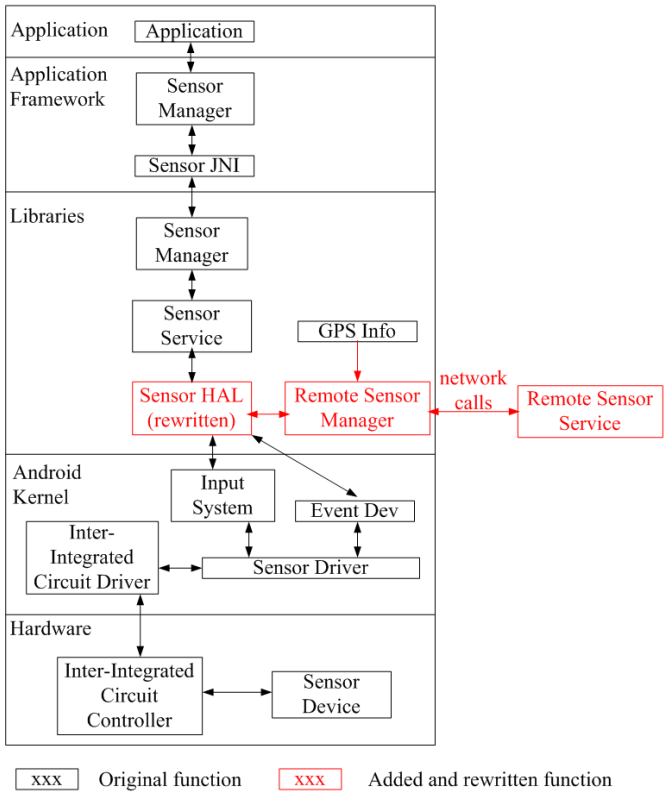


Figure 7. Changed sensor framework in application virtualization module

Fig. 7 shows the new sensor framework for terminal virtualization. In this framework, the original Sensor Hardware Abstraction Layer (HAL) is rewritten. Some new functions are added to the new HAL, including determining if remote sensor services should be accessed, sending the request to remote sensors and receiving the response and updating information. The new block, remote sensor manager, is the interface between the local system and remote sensor services. It is responsible for exchanging the sensor information, including a remote sensor list for the local system, and the terminal's Global Position System (GPS) information for remote sensor services to acquire the terminal's position and remote sensor's information corresponding to the terminal's position.

D. Computation Virtualization Implementation

In the computation virtualization module, we plan to take different approaches according to the type of tasks. For example, for the tasks requiring sophisticated computing power defined in advance, RPC/RMI method will be used; for the independent tasks undefined in advance, virtual machine migration will be used.

Offloading the computation tasks to remote execution can be a solution to go over the limitations of mobile terminal's computing power and can be seen as a way for extending terminal battery. Remote execution leverages the high computation capacity of the server to extend the poor one of the terminal and battery through energy savings. Similarly, thanks to the increased memory of the remote server, it is possible to provide a large set of applications to the mobile user, which are difficult to be run on the terminals.

However, computation offloading is energy efficient only under various conditions. It is vital to look at what happened in context. So, the computation offloading algorithm is very important in the computation virtualization module.

Maui [18] and CloneCloud [19] can be considered as solutions. In Maui, Microsoft's researchers presented a fine-grained solution to reduce programmers work by partitioning the application code automatically, deciding at runtime which methods should be offloaded to be remotely executed. Maui formulates the problem as an Integer Linear Programming (ILP). A comparison of the energy consumption of three applications (face recognition application, video game and chess game) pointed out that energy savings from 27% up to 80% can be achieved. However, this approach works only on Microsoft Windows OS and requires only one server serving each application, which is not scalable to handle many applications.

In CloneCloud, a more coarse-grained offloading approach is proposed that offloading is performed by a modified Dalvik VM automatically. Meanwhile, this approach requires pre-processing on the client side, which can also lead to excessive network traffic from the cloud network to the terminal.

These two approaches do not provide QoE guarantees for mobile users in case of computation offloading. Moreover, the impact of mobile environment in the matter of quality of radio interface according to the position of the end user in the cell has not been considered in these solutions.

A solution named Mobile Application's Offloading (MAO) algorithm was proposed [25], which considers a combination of multiple constraints including CPU, State of Charge (SoC) of the battery, and bandwidth capacity. MAO also evaluates the performance of CPU intensive and I/O interactive applications. The objective is to focus on job offloading, which is a convenient way to achieve energy consumption optimization if certain constraints are satisfied.

The specificities of MAO algorithm with respect to other two offloading algorithms are points out by Table I. "Network Conditions" refers to the fluctuated quality of the radio interface and "User Mobility" refers to the position of the end user within the cell. Compared with Maui and CloneCloud, MAO considers all five aspects, such as QoE, user mobility and SoC of battery. So, it is more applicable to mobile terminal with limited battery.

TABLE I. COMPARISON OF COMPUTATION OFFLOADING ALGORITHMS

Aspects	QoE satisfaction	Network Conditions	User Mobility	CPU	SoC
Maui	No	Yes	No	Yes	No
CloneCloud	No	Yes	No	Yes	No
MAO	Yes	Yes	Yes	Yes	Yes

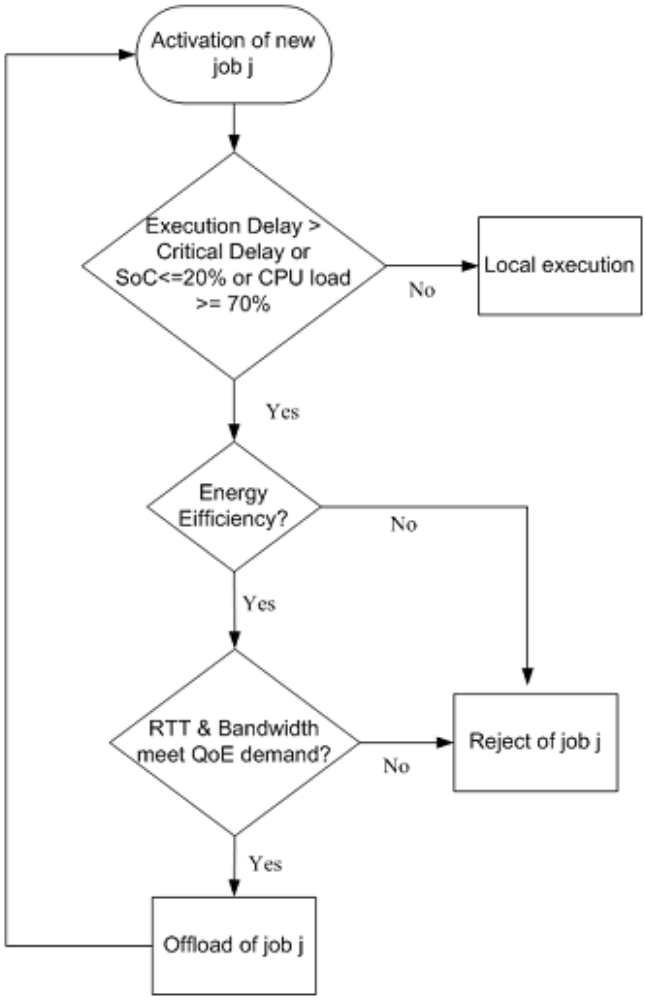


Figure 8. The updated MAO algorithm flowchart

Based on the MAO algorithm, we added one aspect, the CPU load, and embodied QoE as Round-Trip Time (RTT) to the server and access network bandwidth. The updated flowchart is shown in Fig. 8.

E. Security Function Implementation in Management Module

Because the terminal virtualization needs to transport some vital information, such as VM, part of application, users' identification related, security issues are very important to avoid exposing it to the network directly.

To implement the security function in the management module, the data communication should to be encrypted to ensure integrality and confidentiality.

Therefore, considering these characteristics in this security case, a solution based on public-key cryptography is proposed [26], also known as asymmetric cryptography.

In this solution, the cloud platform is employed as the server-side to implement the key management and server-side encryption/decryption function.

When the terminal communicates with the server with security transport, the specific steps are described below. Fig. 9 and Fig. 10 show the steps of the security function.

Uplink (from the terminal to the server)

Step 1: the server in the cloud platform generates public key and private key and sends the public key to the Operation Administration and Maintenance (OAM).

Step 2: when the terminal needs to communicate safely with the server, the terminal gets the public key of the server from OAM. Then the terminal first generates a MD5 [27] of the sending information for the server to validate the integrality of it. Next, the terminal encrypts the sending information and MD5 digest using the public key of the server.

Step 3: the terminal sends the encrypted packet to the server through the network.

Step 4: the server decrypts the packet received using its private key;

Step 5: the server generates its own MD5 of the decrypted information and compares it with the MD5 received from the terminal to determine the integrality of the packet received. If the two MD5s are equal, the information received is reliable. Otherwise, the information has been tampered and should be discarded and retransmitted.

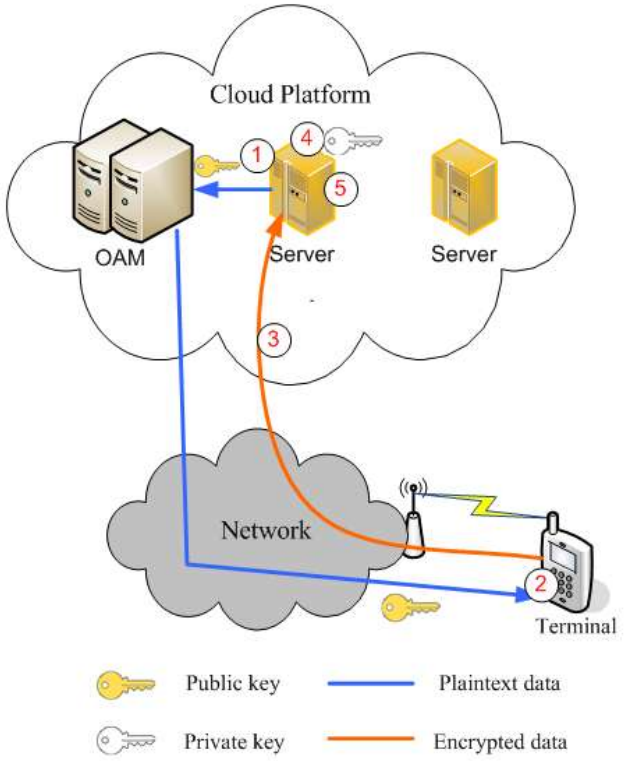


Figure 9. The uplink steps of the security function

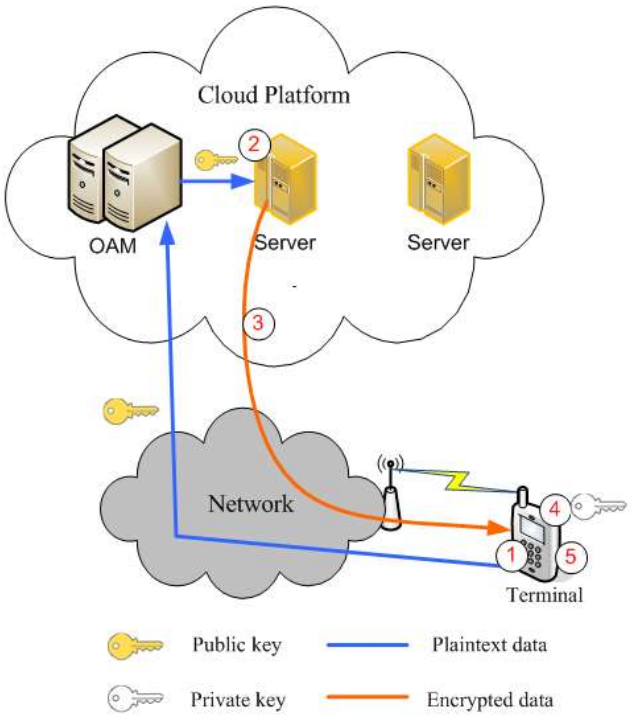


Figure 10. The downlink steps of the security function

Downlink (from the server to the terminal)

Step 1: the terminal generates public key and private key after it boots and sends the public key to the OAM.

Step 2: when the server needs to communicate safely with the terminal, the server gets the public key of the terminal from OAM. Then the server first generates a MD5 of the sending information for the terminal to validate the integrality of it. Next, the server encrypts the sending information and MD5 digest using the public key of the terminal.

Step 3: the server sends the encrypted packet to the terminal through the network.

Step 4: the terminal decrypts the packet received using its private key;

Step 5: the terminal generates its own MD5 of the decrypted information and compares it with the MD5 received from the server to determine the integrality of the packet received. If the two MD5s are equal, the information received is reliable. Otherwise, the information has been tampered and should be discarded and retransmitted.

VI. Conclusion and Future Work

Terminal virtualization has overlapped with some areas, such as mobile peer-to-peer computing, application partitioning, and context-aware computing, but it still has its own unique challenges. There is still a long way to go before terminal virtualization is widely used.

In this paper, we try to build up a unified framework to cover all aspects of terminal virtualization using current technologies and platforms. We analyze some aspects of current terminal virtualization, highlight the motivation for it, and present its functions, applications and some challenges. And on this basis, we proposed a terminal virtualization framework for mobile services. In this framework, four processing modules and one management module are employed to handle the resource requests from apps and shield the details for accessing cloud services. Then we gave some implementation details of these modules.

In the future work, we shall consider analyzing the performance of the prototype in this framework, and give some improved recommendations on the performance.

References

[1] T. Zheng and S. Dong, "Terminal Virtualization for Mobile Services," ICNS 2015, pp. 31-37.

[2] "Cisco Visual Networking Index: Forecast and Methodology, 2014–2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html, May 2015.

[3] "Mobile Cloud Applications & Services," Juniper Research, 2010.

[4] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: a Measurement Study and Implications for Network Applications," Proc. ACM IMC, 2009, pp. 280-293.

[5] P. Shu, F Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, "eTime: Energy-Efficient Transmission between Cloud and Mobile Devices," Proc. IEEE INFOCOM, 2013, pp. 195-199.

[6] R. Chow et al., "Authentication in the Clouds: a Framework and Its Application to Mobile Users," Proc. ACM Cloud Computing Security Wksp. 2010, pp. 1-6.

[7] D. Huang, Z.Zhou, L. Xu, T. Xing, and Y. Zhong, "Secure Data Processing Framework for Mobile Cloud Computing," Proc. IEEE INFOCOM, 2011, pp. 614-618.

[8] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures," Proc. Wksp. Privacy Enhancing Technologies, 2006, pp. 393-412.

[9] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," Proc. IEEE ICDCS 2002, pp. 217-226.

[10] R. Balan, M. Satayanarayanan, S. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," Proc. ACM Mobisys, 2003, pp. 273-286.

[11] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," Masters Thesis, Carnegie Mellon University, 2009.

[12] G. Huerta-Canepa, and D. Lee, "A virtual cloud computing provider for mobile devices," Proc. ACM MCS 2010, article No. 6.

[13] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," Proc. ACM Mobisys, 2010, pp. 59-79.

[14] D. C. Doolan, S. Tabirca, and L.T. Yang, "Mmpi a message passing interface for the mobile environment," Proc. ACM Mobisys, 2008, pp. 317-321.

[15] M. Kristensen, "Scavenger: transparent development of efficient cyber foraging applications," Proc. IEEE PerCom, 2010, pp. 217-226.

[16] C. Clark, et al., "Live migration of virtual machines," Proc. of the 2nd conference on Symposium on Networked Systems Design & Implementation, USENIX Association, 2005, vol 2, pp. 273-286.

[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM –based Cloudlets in Mobile Computing," IEEE Pervasive Computing, vol. 8, no. 4, 2009.

[18] E. Cuervo et al., “Maui: Making Smartphones Last Longer with Code Offload,” Proc. ACM MobiSys, 2010, pp. 49-62.

[19] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “Clonecloud: Elastic Execution Between Mobile Device and Cloud,” Proc. ACM EuroSys, 2011, pp. 301-314.

[20] D. Huang, X. Zhang, M. Kang, and J. Luo, “MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication,” Proc. IEEE SOSE, 2010, pp. 27-34.

[21] http://www.yunos.com, YunOS, November 2015.

[22] T. Zheng and D. Gu, “Traffic Offloading Improvements in Mobile Networks,” ICNS 2014, pp. 116-121.

[23] http://www.ccnx.org, CCNx, November 2015.

[24] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” Proc. ACM SIGCOMM '98, pp. 56-67.

[25] A. Ellouze, M. Gagnaire, and A. Haddad, “A Mobile Application Offloading Algorithm for Mobile Cloud Computing,” Proc. IEEE MobileCloud 2015, pp. 34-40.

[26] http://www.merkle.com/1974/, R. C. Merkle, November 2015.

[27] http://en.wikipedia.org/wiki/MD5, November 2015.

Chronomorphic Programs: Runtime Diversity Prevents Exploits and Reconnaissance

Scott E. Friedman, David J. Musliner, and Peter K. Keller
Smart Information Flow Technologies (SIFT)
Minneapolis, USA
email: {sfriedman,dmusliner,pkeller}@sift.net

***Abstract*—In Return Oriented Programming (ROP) attacks, a cyber attacker crafts an exploit from instruction sequences already contained in a running binary. ROP attacks are now used widely, bypassing many cyber defense mechanisms. While previous research has investigated software diversity and dynamic binary instrumentation for defending against ROP, many of these approaches incur large performance costs or are susceptible to Blind ROP attacks. We present a new approach that automatically rewrites potentially-vulnerable software binaries into *chronomorphic* binaries that change their in-memory instructions and layout repeatedly, at runtime. We describe our proof of concept implementation of this approach, discuss its security and safety properties, provide statistical analyses of runtime diversity and reduced ROP attack likelihood, and present empirical results that demonstrate the low performance overhead of actual chronomorphic binaries.**

Keywords-cyber defense; software diversity; self-modifying code.

I. Introduction

In the old days, cyber attackers only needed to find a buffer overflow or other vulnerability and use it to upload their exploit instructions, then make the program execute those new instructions. To counter this broad vulnerability, modern operating systems enforce "write XOR execute" (W⊕X) defenses: that is, memory is marked as either writable or executable, but not both. So exploit code that is uploaded to writable memory cannot be executed. Not surprisingly, attackers then developed a more sophisticated exploit method.

Computer instruction sets are densely packed into a small number of bits, so accessing those bits in ways that a programmer did not originally intend can yield *gadgets*: groups of bits that form valid instructions that can be strung together by an attacker to execute arbitrary attack code from an otherwise harmless program. Known as *Return Oriented Programming* (ROP), these types of cyber exploits have been effective and commonplace since the widespread deployment of W⊕X defenses. This paper extends our previous research on runtime program security to prevent ROP exploits [1].

Software with a single small buffer-overflow vulnerability can be hijacked into performing arbitrary computations using ROP-like code-reuse attacks [2], [3]. Hackers have even developed *ROP compilers* that build the ROP exploits automatically, finding gadgets in the binary of a vulnerable target and stringing those gadgets together to implement the attacker's code [4], [5]. And to counteract various software diversity defenses that try to move the gadgets around, so that a previously-compiled ROP attack will fail, attackers have developed *Blind ROP* (BROP) attacks that perform automated reconnaissance to find the gadgets in a running program [6].

This paper presents a fully automated approach for transforming binaries into *chronomorphic* binaries that diversify themselves during runtime, throughout their execution, to offer strong statistical defenses against code reuse exploits such as ROP and BROP attacks. The idea is to modify the binary so that all of the potentially-dangerous gadgets are repeatedly changing or moving, so that even a BROP attack tool cannot accumulate enough information about the program's memory layout to succeed.

In the following sections, we discuss related research in this area (Section II), we outline the threat of ROP exploits (Section III), we describe how our prototype Chronomorph tool converts regular binaries into chronomorphic binaries (Section IV), and we review its present limitations. We then describe an analysis of the safety and security of the resulting chronomorphic binaries, and performance results on early examples (Section V). We conclude with several directions for future work, to harden the tool and broaden its applicability (Section VI).

II. Related Work

Various defense methods have been developed to try to foil code reuse exploits such as ROP and BROP. Some of defenses instrument binaries to change their execution semantics [7] or automatically filter program input to prevent exploits [8]; however, these approaches require process-level virtual machines or active monitoring by other processes. Other approaches separate and protect exploitable data (e.g., using shadow stacks [9]), but such approaches incur comparatively high overhead.

To reduce overhead and maintain compatibility with existing operating systems and software architectures, many researchers have focused on lightweight, diversity-based techniques to prevent code reuse exploits. For example, Address Space Layout Randomization (ASLR) is common in modern operating systems, and loads program modules into different locations each time the software is started. However, ASLR does not randomize the location of the instructions *within* loaded modules, so programs are still vulnerable to ROP attacks [10]. Some diversity techniques modify the binaries themselves to make them less predictable by an attacker. For example:

- Compile-time diversity (e.g., [11]) produces semantically equivalent binaries with different structures.

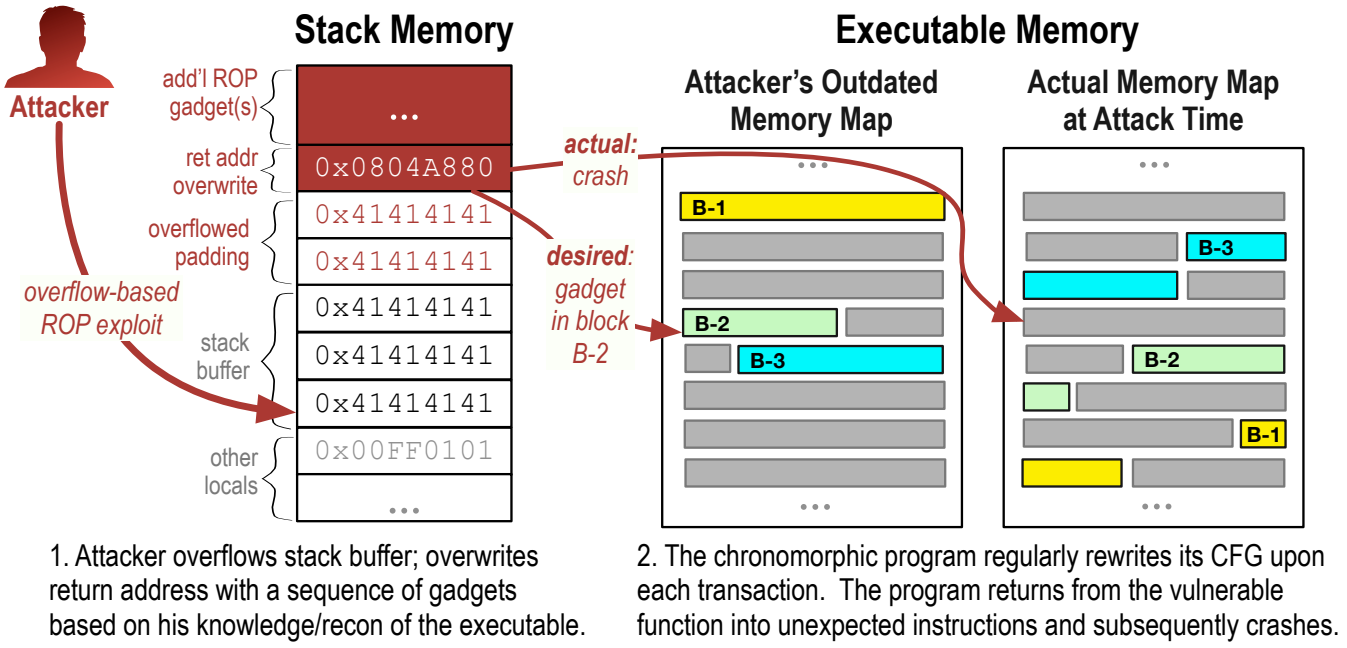


Figure 1. A traditional ROP attack– or blind ROP attack– is thwarted by the runtime diversity of a chronomorphic program.

- Offline code randomization (e.g., [12]) transforms a binary on disk into a functionally equivalent variant with different bytes loaded into memory.
- Load-time code randomization (e.g., [13], [14]) makes the binary load blocks of instructions at randomized addresses.

These diversity-based approaches incur comparatively lower overhead than other ROP defenses and they offer statistical guarantees against ROP attacks.

Unfortunately, these compile-time, offline, and load-time diversity defenses are still susceptible to BROP attacks that perform runtime reconnaissance to map the binary and find gadgets [6]. So, even with compile-time, offline, or load-time diversity, software that runs for a significant period of time without being reloaded (e.g., all modern server architectures) is vulnerable. Some ROP defenses modify the operating system to augment diversity [15], [16], but by nature they do not work on existing operating systems.

Another recent approach uses compile-time diversity in tandem with hardware-based enforcement mechanisms that prevents adversaries from reading any code [17]. This protects against memory disclosure— and thereby prevents ROP and BROP attacks— but like the above techniques, this requires modifying the underlying operating system or hardware.

Some runtime techniques clone executable elements in memory and toggle between them during runtime, so the attacker is unaware of which memory layout is executing; however, the diversity factor is not as high as the above approaches. Another recent approach combines execution-time switching with runtime diversification [18] by instrumenting all call, return, and jump instructions. On these instrumented instructions, execution may randomly switch between executable copies while the other copy is diversified by fine-grained ASLR. While this approach prevents varieties of ROP attacks, it incurs significant runtime overhead due to a dynamic binary instrumentation framework, and it doubles the size of the binary due to executable memory cloning.

Other tools such as DynInst[1] rewrite the binary to add runtime libraries and instrumentation, but this instrumentation consumes substantially higher disk space, memory footprint, or performance overhead.

Unlike the above diversity-based protection techniques, chronomorphic programs only utilize a single instance of the program in memory at any time, making them more suitable for embedded or memory-constrained systems. Chronomorphic programs will diversify themselves *throughout program execution* to statistically prevent code reuse attacks, even if the attacker knows the memory layout. The only runtime costs are incurred when actually morphing the program; when not morphing, the program executes (almost) its original instructions. Furthermore, the costs of the morphing behavior can be adjusted and controlled to achieve desired performance levels in the face of changing threat levels, rates of adversary-provided input, etc. Unlike other approaches, our prototyped chronomorphic programs run on existing hardware and operating systems, making them suitable for legacy systems.

III. Threat Model

Here we describe the setting for chronomorphic programs— including assumptions about the target program, the target system, and the attacker— and we illustrate this setting in

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

Figure 1. The chronomorphic defense against code reuse attacks assumes the following of the targeted system and its adversaries:

- The target program has a vulnerability that permits the adversary to hijack the program's control flow.
- The target host uses write-xor-execute (W⊕X) permissions on its memory pages.
- The target operating system contains standard functions to modify W⊕X memory permissions (e.g., `mprotect` in Linux and `VirtualProtect` in Windows), which we describe later.
- The adversary cannot influence the offline operations that create the chronomorphic binary and its morph table from a standard executable (see Figure 2).
- The adversary may have access to the target program's source code and to the original (non-chronomorphic) off-the-shelf executable.
- The adversary may have *a priori* knowledge of the program's in-memory code layout at *any* point in execution, unlike other ROP defenses (e.g., [11], [12], [13], [14], [17]), due to cyber reconaissance (e.g., [6]) or *runtime information leaks* whereby timing data, cache data, and side channel data is available to the attacker.
- The adversary must interact with the target program (e.g., execute its in-memory code) in order to observe its in-memory code layout or conduct an exploit.

Under these assumptions, the adversary may successfully exploit the target program over multiple transactions (e.g., server requests), *provided the target program does not change its exploitable memory layout before or after those transactions*.

Chronomorphic programs foil cyber-attacks that rely on consistency of a program's memory layout— including code reuse attacks like ROP— since the memory layout changes during execution. As described below, these memory changes should occur early in the processing of each transaction. Figure 1 illustrates an attempt of a classic stack-overflow-based ROP exploit on a chronomorphic executable. The attacker overflows a stack buffer to overwrite the return address with one or more gadget addresses that he may have learned by analyzing the executable on disk or by exploiting the runtime information leaks mentioned above.

The chronomorphic program rewrites itself regularly throughout execution (Figure 1, right), so the attacker's knowledge of the program is outdated. When the program returns into the chain of gadgets written by the attacker, the program executes different instructions than those intended by the attacker. The program promptly crashes without executing the exploit.

The empty, unlabeled area (i.e., block relocation space) could be comprised of invalid, unexploitable instructions that quickly cause a crash, or alternatively, nop-slides into an invocation of alarms or forensic analysis functions that terminate with an error signal, consistent with other software "booby trapping" approaches [19].

Chronomorphic programs diversify themselves based on random selections from their *morph tables*, which describe how the program can be changed during execution. This means that if the morph table is aware to the attacker *and* the target host's `random` operation is perfectly predictable, then the attacker can predict the next configuration assumed by the chronomorphic program and thereby conduct a successful code reuse exploit. Consequently, even if the morph table is acquired and decrypted by the adversary, they must perfectly predict— or somehow influence— the host's randomization, which would already constitute a deep intrusion of the target system.

We next describe how we create chronomorphic programs from off-the-shelf executables, and how these programs diversify themselves throughout the course of their execution.

IV. Approach

The Chronomorph approach requires changing machine code at runtime, a technique known as *self-modifying code* (SMC). Using SMC, Chronomorph must preserve the functionality of the underlying program (i.e., maintain semantics), maximize diversity over time, and minimize performance costs.

Any SMC methodology requires a means to change the permissions of the program's memory (i.e., temporarily circumvent W⊕X defense) to modify the code and then resume its execution. Different operating systems utilize different memory protection functions: Linux's `mprotect` and Windows' `VirtualProtect` have different signatures, but both can temporarily change program memory permissions from executable to writable, and back again, during program execution. In this paper, we describe Chronomorph in a 32-bit Linux x86 setting.

Our approach automatically constructs chronomorphic binaries from normal third-party programs with the following enumerated steps, also illustrated in Figure 2:

Offline:

1) Transform the executable to inject the Chronomorph SMC runtime that invokes `mprotect` and rewrites portions of the binary during execution. This produces a *SMC binary* with SMC functions that are disconnected from the program's normal control flow.
2) Analyze the SMC binary to identify potentially-exploitable sequences of instructions (i.e., gadgets).
3) Identify relocatable gadgets and transform the SMC binary to make those gadgets relocatable.
4) Compute instruction-level, semantics-preserving transforms that denature non-relocatable gadgets and surrounding program code.
5) Write the relocations and transforms to a *morph table* outside the chronomorphic binary.
6) Inject *morph triggers* into the SMC binary so that the program will morph itself periodically. This produces the *chronomorphic binary*.

Online:

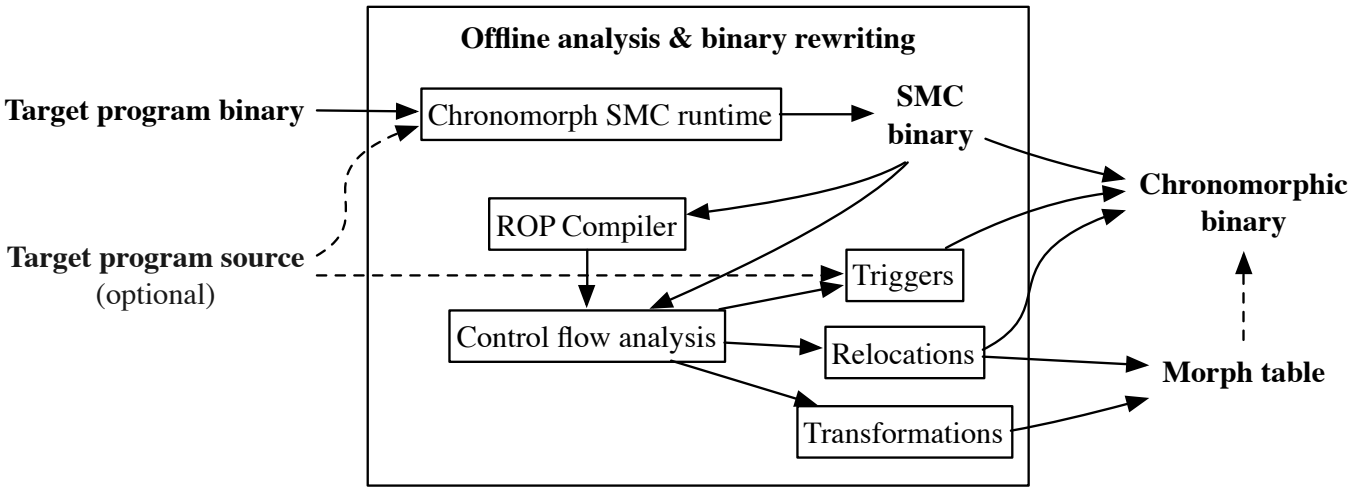


Figure 2. Chronomorph converts a third-party program into a chronomorphic binary.

7) During program runtime, diversify the chronomorphic binary's executable memory space by relocating and transforming instructions without hindering performance or functionality.

We have implemented each step in this process and integrated third-party tools including a ROP compiler [5], the Hydan tool for computing instruction-level transforms [20], and the open-source `objdump` disassembler. We next describe each of these steps in this process, including the research challenges and the strategy we employ in our Chronomorph prototype implementation. We note relevant simplifying assumptions in our prototype, and we address some remaining research challenges in Section VI.

A. Injecting SMC morphing functionality

Before the Chronomorph tool can analyze the binary and compute transformations, it must inject the Chronomorph SMC runtime, which contains functions for modifying memory protection (e.g., `mprotect`), writing byte sequences to specified addresses, and reading the morph table from outside the binary. These Chronomorph functions may *themselves* contain gadgets and have runtime diversification potential, so the SMC-capable binary that includes these functions is the input to the subsequent offline analyses, including ROP compilation.

We identified three ways of automatically injecting the Chronomorph runtime code, based on the format of the target program.

1) Link the target program's source code against the compiled Chronomorph runtime. This produces a dynamically- or statically-linked SMC executable. This is the simplest solution, and the one used in our experiments, but source code may not always be available.
2) Rewrite a statically-linked binary by extending its binary with a new loadable, executable segment containing the statically-linked Chronomorph runtime. This produces a statically-linked SMC executable.
3) Rewrite a dynamically-linked binary by adding Chronomorph procedures and objects to an alternative procedure linkage table (PLT) and global object table (GOT), respectively, and then extend the binary with a new loadable, executable segment containing the dynamically-linked Chronomorph runtime. This produces a dynamically-linked SMC executable.

All three of these approaches inject the self-modifying Chronomorph runtime, producing the *SMC binary* shown in Figure 2. At this point, the self-modification functions are not yet invoked from within the program's normal control flow, so we cannot yet call this a chronomorphic binary.

B. Identifying exploitable gadgets

As shown in Figure 2, the Chronomorph offline analysis tool includes a third-party ROP compiler [5] that automatically identifies available gadgets within a given binary and creates an exploit of the user's choice (e.g., execute an arbitrary shell command) by compiling a sequence of *attack gadgets* from the available gadgets, if possible. The Chronomorph analysis tool runs the ROP compiler against the SMC binary, finding gadgets that span the entire executable segment, including the Chronomorph SMC runtime.

The ROP compiler prioritizes Chronomorph's diversification efforts as follows, to allocate time and computing resources proportional to the various exploitation threats within the binary:

- Attack gadgets have the highest priority. The chronomorphic binary should address these with its highest-diversity transforms.
- Available gadgets (i.e., found by the ROP compiler but not present in an attack sequence) have medium priority. These too should be addressed by high-diversity transforms, within acceptable performance bounds.
- Instructions that have not been linked to an available gadget have the lowest priority, but should still be diversified

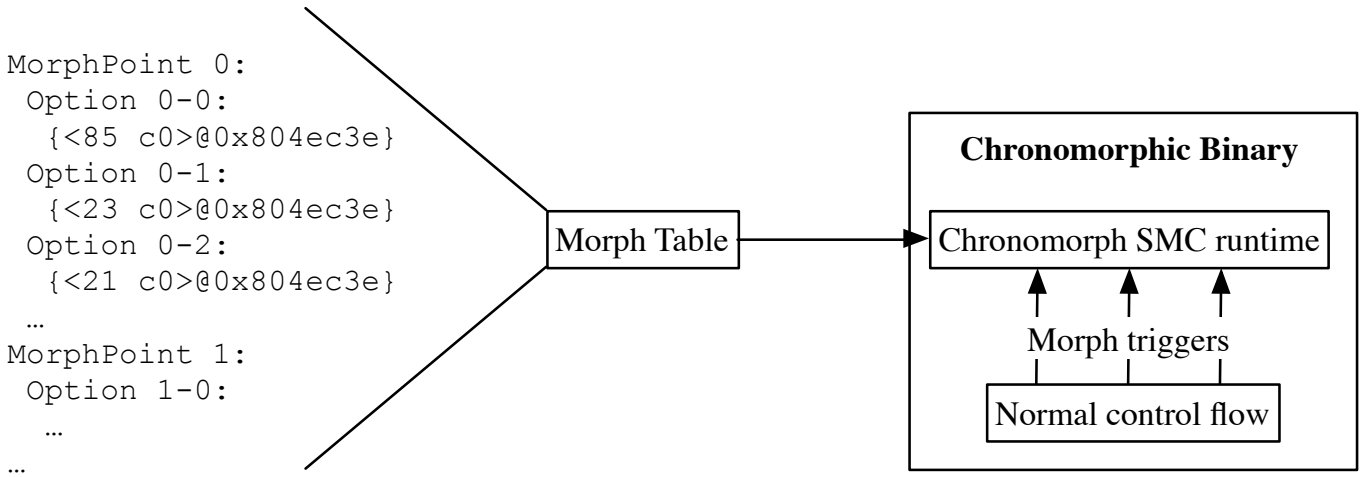


Figure 3. The resulting chronomorphic binary and its interaction with the morph table.

when feasible. Since zero-day gadgets and new code-reuse attack strategies may arise after transformation time, this diversification offers additional security.

Our approach attempts all transformations possible, saving more costly transformations, e.g., dynamic block relocations, for the high-risk attack gadgets. The ROPgadget compiler [5] currently used by Chronomorph may be easily replaced by newer, broader ROP compilers, provided the compiler still compiles attacks and reports all available gadgets. Also, a portfolio approach may be used, running a variety of ROP compilers and merging their lists of dangerous gadgets.

C. Diversity with relocation

We may not be able to remove a high-risk gadget entirely from the executable, since its instructions may be integral to the program's execution; however, the chronomorphic binary can relocate it with high frequency throughout execution, as long as it preserves the control flow.

Relocation is the highest-diversity strategy that Chronomorph offers. Chronomorph allocates an empty *block relocation space* in the binary, reserved for gadget relocation. Whenever the chronomorphic binary triggers a morph, it shuffles relocated blocks to random locations in the block relocation space and repairs previous control flow with recomputed `jmp` instructions to the corresponding location in the block relocation space.

For each high-risk attack gadget, Chronomorph performs the following steps to make it relocatable during runtime:

1) Compute the *basic block* (i.e., sequence of instructions with exactly one entry and exit point) that contains the gadget.
2) Relocate the byte sequence of the gadget's basic block to the first empty area in the block relocation space.
3) Write a `jmp` instruction from the head of the basic block to the new address in the block relocation space.
4) Write `nop` instructions over the remainder of the gadget's previous basic block, destroying the gadgets.
5) Write the block's byte sequence and the address of the new `jmp` instruction to the morph table.

The morph table now contains enough information to place the gadget-laden block anywhere in the block relocation space and recompute the corresponding `jmp` instruction accordingly.

Intuitively, diversity of the binary increases with the size of the block relocation space. For a single gadget block g with byte-size $|g|$, and block relocation space of size $|b|$, relocating g adds $V(g,b) = 1 + |b| - |g|$ additional program variants.

If we relocate multiple gadget blocks $G = \{g_0, ..., g_{|G|-1}\}$, then we add the following number of variants:

$$V(G,b) = |G| + \prod_{i=0}^{|G|-1} (|b| - \sum_{j=0}^{i} |g_j|). \quad (1)$$

The probability of guessing all of the relocated gadgets' addresses is therefore $1/V(G,b)$, which diminishes quickly as the chosen size of the block relocation space increases.

Our Chronomorph prototype has the following constraints for choosing gadget blocks for relocation:

- Relocated blocks cannot contain a `call` instruction. When a `call` instruction is executed, the subsequent instruction's address is pushed onto the stack, and if the calling block is then relocated, execution would return into an arbitrary (incorrect) spot in the block relocation space.
- Relocated blocks must be at least the size of the `jmp` to the block relocation space, so that Chronomorph has room to write the `jmp`.
- Relocated blocks must end in an indirect control flow (e.g., `ret`) instruction; otherwise, we would have to recompute the control flow instruction at the block's tail at every relocation. Empirically, the vast majority of these blocks end in `ret`.
- Relocated gadgets cannot span two blocks.

In the conclusion of this paper we discuss some potential improvements that would remove some of these constraints.

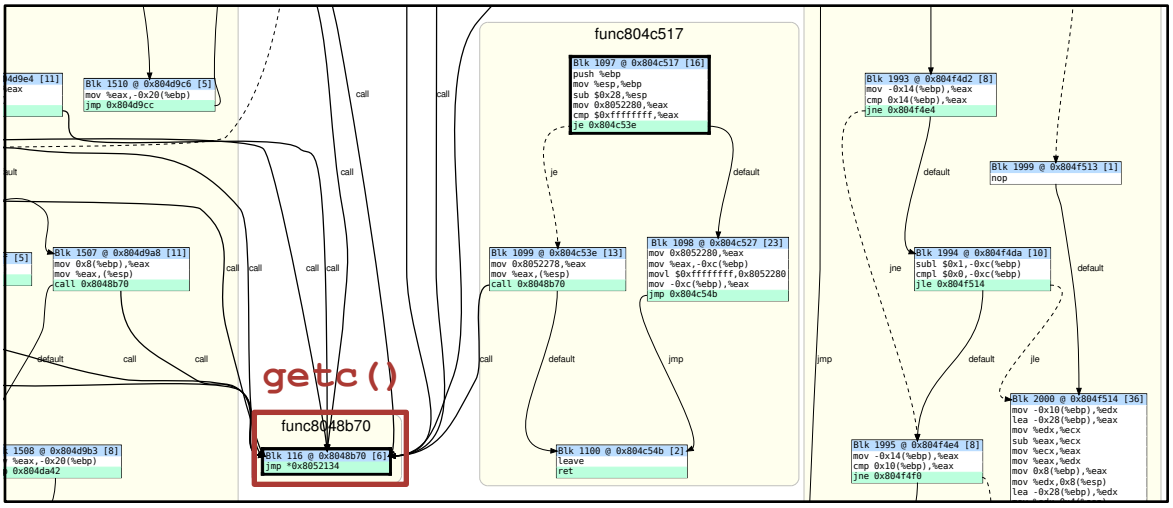


Figure 4. Small portion of a control flow graph (CFG) automatically created during Chronomorph's offline analysis. Shaded regions are functions, instruction listings are basic blocks, and edges are control flow edges.

D. Diversity with in-place code randomizations

Chronomorph uses *in-place code randomization* (IPCR) strategies to randomize non-relocated instructions [12]. IPCR performs narrow-scope transformations without changing the byte-length of instruction sequences.

At present, Chronomorph uses two IPCR strategies to compute transformations. The first, *instruction substitution* (IS), substitutes a single instruction for one or more alternatives. For example, comparisons can be performed in either order, `xor`'ing a register with itself is equivalent to `mov`'ing or `and`'ing zero, etc. These instructions have the same execution semantics, but they change the byte content of the instruction, so unintended control flow instructions (e.g., `0xC3` = `ret`) are potentially transformed or eliminated. A single IS adds as many program variants as there are instruction alternatives.

Another IPCR strategy, *register preservation code reordering* (RPCR) reorders the `pop` instructions before every `ret` instruction of a function, and also reorders the corresponding `push` instructions at the function head to maintain symmetry. A register preservation code reordering for a single function adds as many variants as there are permutations of `push` or `pop` instructions.

Importantly, RPCR changes the layout of a function's stack frame, which may render it non-continuable. For instance, if control flow enters the function and it preserves register values via `push`'ing, and then the chronomorphic binary runs RPCR on the function, it will likely `pop` values into unintended registers when it continues the function, which will adversely affect program functionality.

Any stack-frame diversity method such as RPCR should only be attempted at runtime if execution cannot *continue* or *re-enter* the function, e.g., from an internal `call`, after a SMC morph operation. We enforce this analytically with control flow graph (CFG) analysis: if execution can continue within a function f from the morph trigger (i.e., if the morph trigger is *reachable* from f in the CFG), the stack frame of f should not be diversified. Stack frame diversification is a valuable tool for ROP defense, but it requires these special considerations when invoked during program execution.

E. Writing and reading the morph data

The morph table is a compact binary file that accompanies the chronomorphic binary, as shown in Figure 3. The morph table binary represents packed structs: `MorphPoint` structs with internal `MorphOption` byte sequences. Each `MorphPoint` represents a decision point (i.e., an IS or RPCR opportunity) where any of the associated `MorphOption` structs will suffice. Each `MorphPoint` is stateless (i.e., does not depend on the last choice made for the `MorphPoint`), and independent of any other `MorphPoint`, so random choices are safe and ordering of the morph table is not important.

The relocation data is a separate portion of the morph table, containing the content of relocatable blocks alongside their corresponding `jmp` addresses. Like IPCR operations, relocations are stateless and independent, provided the Chronomorph runtime does not overlay them in the block relocation space.

Intuitively, the morph table cannot reside statically inside the binary as executable code, otherwise all of the gadgets would be accessible.

F. Injecting morph triggers

We have described how Chronomorph injects SMC capabilities into third-party executables and its diversification capabilities, but Chronomorph must also automatically connect the Chronomorph runtime into the program's control flow to induce diversification of executable memory during runtime.

The injection of these *morph triggers* presents a trade-off: morphing too frequently will unnecessarily degrade program performance; morphing too seldom will allow wide windows of attack. Ideally, morphing will happen at the speed of input,

e.g., once per server request or user input (or some modulo thereof). The location of the morph trigger(s) in the program's control flow ultimately determines morph frequency.

Figure 4 shows a portion of the CFG for the program used in our experiment, calling out the `getc()` input function. Chronomorph can inject calls to the SMC runtime at these input points, or at calling functions with stack-based buffers (which are more likely to contain vulnerabilities through which a ROP attack would begin).

Chronomorph also includes an interface for the application developer to add a specialized `MORPH` comment in the source code, which is replaced by a morph trigger during the rewriting phase.

G. Runtime diversification

A chronomorphic binary executes in the same manner as its former non-chronomorphic variant, except when the morph triggers are invoked.

When the first morph trigger is invoked, the Chronomorph runtime loads the morph table and seeds its random number generator. All morph triggers induce a complete SMC diversification of the in-process executable memory according to IPCR and relocation data in the morph table:

1) The block relocation space is made writable with `mprotect`.
2) Relocated basic blocks in the block relocation space are overwritten with `nop` instructions.
3) Each relocatable block is inserted to a random block relocation space address, and its `jmp` instruction (where the block used to be in the program's original CFG) is rewritten accordingly.
4) The block relocation space is made executable.
5) Each `MorphPoint` is traversed, and a corresponding `MorphOption` is chosen at random and written. Each operation is surrounded by `mprotect` calls to make the corresponding page writable and then executable. Future work will group `MorphPoints` by their address to reduce `mprotect` invocations, but our results demonstrate that the existing performance is acceptable.

We have described how to create chronomorphic binaries from off-the-shelf binaries, and we have described how chronomorphic binaries perform runtime diversification. Next we discuss important safety and correctness considerations for chronomorphic programs, since runtime rewriting has implications for concurrency, continuation, and reentrancy.

H. Multithreading

In a traditional multi-threaded setting, multiple threads use the same executable memory. This means that one thread could diversify the function, block, or even the instruction that another thread is executing.

Without additional protections such as thread synchronization, many runtime diversification strategies are unsafe in a multithreaded setting:

- Block relocation is unsafe if another thread is executing a relocatable block.
- Reordering a function's `push` and `pop` register preservation instructions is unsafe if another thread is executing the function, since the thread may `pop` values into the wrong registers.
- Reordering instructions *within* a block (e.g., [12]) is unsafe if another thread is executing the block.

Injecting thread synchronization around diversifiable regions in the chronomorphic program's morph table would prevent these unsafe operations, but this might be costly, since— as we demonstrate in our experiments— there are many diversifiable regions in even the smallest binaries.

For the above reasons, our prototype tool only supports creating chronomorphic binaries for single-threaded execution, but we discuss some possible avenues for multi-threaded chronomorphic programs in our discussion of future work.

I. Guaranteeing safe continuation

Runtime diversity can invalidate stack frame integrity and return address correctness if not properly constrained. These are important considerations for guaranteeing *function continuation* (i.e., resuming execution of a function after returning from another function) within a chronomorphic program. We discuss two continuation pitfalls and our approach for avoiding them.

Reordering `push` and `pop` instructions, are effective methods of foiling ROP attacks [12]; however, performing these operations at runtime will complicate continuation by changing the expected structure of the stack frame: if execution continues in (i.e., returns back into) a function whose `push` and `pop` differ from when execution initially entered, the program will pop the wrong data into the registers, causing a random fault in subsequent execution.

Another consideration is return address correctness. When a program executes a `call` instruction, it will write the address following the `call` onto the stack as the return address, and then transfer control flow to the called function. Suppose that after invoking a `call` instruction and writing the return address, the program triggers a downstream diversification that relocates this upstream `call` instruction. This changes the address that the program *should* return to; however, the old address is still written to the stack, and the program will ultimately return into an undesired location, causing a random fault.

Both of these continuation problems stem from changing executable memory in a way that is inconsistent with presently-written stack memory. There are three general strategies for preserving continuation in light of both of these problems:

1) *Avoidance*: Do not perform `push` and `pop` register preservation reordering, and do not relocate any block containing a `call` instruction.
2) *Rectification*: Rectify stack memory at diversification time by transposing register preservation data (to allow `push` and `pop` reordering) and rewriting return addresses (to allow `call` relocation) of continuable functions.

3) *Reachability*: Do not modify the stack frame (e.g., `push` and `pop` register preservation) or return addresses (e.g., location of `call` instructions) of functions that can plausibly be continued after runtime diversification, using a CFG graph-reachability criterion.

All of these strategies incur a cost. The avoidance strategy reduces diversity by disallowing `push` and `pop` register preservation and disallowing `call` instructions to be relocated. Other strategies such as instruction substitution and the relocation of other non-`call` blocks are still plausible, but this is an unnecessary loss.

The rectification strategy will incur memory overhead to store the data necessary to identify register preservation values and return addresses that have been pushed onto the stack. The time necessary to rectify the stack would be proportional to the depth of the stack, and it could take arbitrarily long to rectify all of the values, e.g., when modifying deep recursive functions.

The reachability strategy reduces diversity potential, but provides more diversity potential than the avoidance strategy. This approach also incurs substantially less overhead than the rectification strategy and permits more diversity than the avoidance strategy, so this is our preferred strategy for chronomorphic programs.

Our prototype tool implements the reachability strategy using a graph-theoretic reachability criteria in the program's CFG. From the CFG, we can infer the set of functions that could reach the runtime diversification function injected into the chronomorphic binary. Our approach recovers the CFG from the binary automatically, using mixed recursive/linear disassembly of the binary, static identification of jump tables, and dynamic tracing to identify indirect control flow (e.g., jump addresses stored as data) [21]. This over-approximates the CFG and ultimately over-approximates the set of functions that *will* be continued after runtime diversification. In a hypothetical worst case, where *every* function in the program can reach the runtime diversification function, the reachability strategy is equivalent to the avoidance strategy described above, which is still preferable to the rectification strategy.

We conducted an experiment with our Chronomorph prototype on a third-party Linux binary to characterize the diversity, ROP attack likelihood, and performance overhead of our Chronomorph approach. We discuss this experiment and its results in the next section.

V. EVALUATION

We tested our prototype tool on small Linux desktop applications, into which we deliberately injected vulnerabilities and gadgets (in the source code). Here, we discuss results for the `dc` (desktop calculator) program.

The original target program, with injected flaws, is easily compromised by our ROP compiler. We also inserted the special `MORPH` comment in the source code, to trigger morphing after each input line was read. After running the prototype Chronomorph system, the new binary operates as described in Figure 3, and cannot be defeated by the ROP compiler. The dynamically-linked version of the original binary is small (47KB), and after our tool has made it chronomorphic (with a block relocation space of 4KB) it is 62KB.

The rewritten binary is currently able to perform approximately 1000 changes to its own code in less than one millisecond, on a standard laptop. When the chronomorphic binary is not rewriting itself, it incurs no additional performance overhead, so the overhead is strictly the product of the time for a complete morph (e.g., one millisecond) and the frequency of morphs, as determined by the injected morph triggers. In our experiments with `dc` performing a short regression test, the chronomorphic version incurred an additional 2% overhead. However, this was an unoptimized version that reloaded the morph table on every morph trigger. For other binaries, overhead will depend heavily on morph trigger placement. Also, a more compute-intensive application might suffer a mild degradation due to cache-misses and branch prediction failures that might not occur in the non-morphing version.

Figure 5 shows two bitmaps illustrating how the binary instructions change in memory, as the program runs. Each pixel of each image represents a single byte of the program's executable code segment in memory. At the top of both images, the gray area is the `nop`-filled block relocation space, with colored segments representing the blocks moved there. Note that the colored segments are in different locations in the two images. Below the gray area, the original binary bits that are never changed remain black, while instructions that are rewritten are shown in different colors, where the red/green/blue values are computed from the byte values and `nop` instructions are gray. Comparison of the images will show that many of the colored areas are different between the images— this clearly shows the broad in-memory diversity induced by the chronomorphic behavior.

We assessed this example's morph table and estimate that it is capable of randomly assuming any one of approximately 10^{500} variants at any given time during execution. The chronomorphic version of the statically-linked target binary ($>$ 500KB) can assume any one of approximately 10^{8000} variants, using about 8ms to perform all of its rewrites. However, those variant counts do not really accurately characterize the probability that a ROP or BROP attack will succeed.

To do that, we must consider how many gadgets the attacker would need to locate, and how they are morphing. The dynamically linked target contains 250 indirect control flow instructions, and two thirds of those potentially risky elements are moved by the block-relocation phase. With the ROPgadget compiler we used for this evaluation, the original application yielded an exploit needing eight gadgets, of which six were subjected to morphing:

- `inc eax ; ret` – relocated.
- `int 0x80` – relocated.
- `pop edx ; ret` – relocated.
- `pop edx ; pop ecx ; pop ebx ; ret` – reordered (6 permutations).
- `pop ebx ; ret` – relocated.
- `xor eax,eax ; ret` – intact.

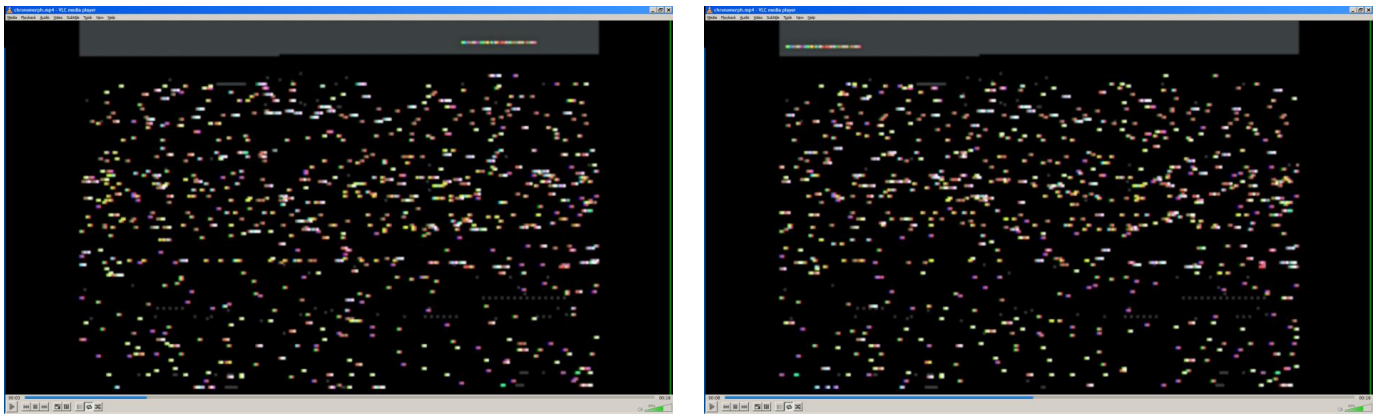


Figure 5. Example memory visualizations illustrating how the executable memory space of the binary changes at runtime.

- `pop eax ; ret` – relocated.
- `move [edx],eax ; ret` – intact.

Five of the gadgets are relocated dynamically within the block relocation space of size $|b|$, and a sixth gadget is rewritten to one of six permutations. As a result, to accurately locate all eight of those gadgets in the chronomorphed binary, a potential ROP attacker would have to pick correctly from approximately $6*|b|^5$ alternatives. For our $|b| = 4$KB example, the probability of a correct guess is approximately $1/10^{18}$, which is extremely unlikely. Needless to say, the ROPgadget exploit was unable to compromise the chronomorphic binary, in thousands of tests. Furthermore, a BROP attack will have no ability to accumulate information about gadget locations, because they change every time a new input is received.

VI. Conclusion and Future Work

We have implemented an initial version of an automatic Chronomorph tool and demonstrated that the resulting chronomorphic binaries are resistant to ROP and BROP attacks and retain their initial functionality. Our automatically-generated chronomorphic binary incurred no runtime overhead during normal operation, and only incurred one millisecond overhead to perform over 1000 sequential rewrites to executable memory during a morph operation.

Our initial version and experimental results demonstrate that chronomorphic programs are feasible: runtime diversity yields real security benefits on existing software running on existing operating systems and hardware. For chronomorphic programs to become widespread and practical, further research must characterize the effect on chronomorphic programs' error reporting capabilities, performance tuning, and reliable binary disassembly for safe code relocation. We prototyped our approach in a 32-bit x86 Linux setting, and scaling to x86_64 would increase the size of the morph table (due to 64-bit addresses), require a x86_64 ROP compiler, and require support for disassembly and binary rewriting to account for x86_64 argument-passing via registers.

Additional high-level research challenges remain for safety and scalability. The system call that allows the chronomorphing code to rewrite executable code is, of course, a dangerous call; if an attacker could locate it and exploit it, he could rewrite the code to do whatever he wants. Therefore, we would ideally like the rewriting/SMC code itself to relocate or transform at runtime; however, the code cannot rewrite itself. We can work around this limitation with a fairly simple trick: we can use two copies of the critical code to alternately rewrite or relocate each other throughout runtime.

Another security concern requiring additional research is ROP attacks *tailored* to chronomorphic programs. If the attacker has full reconnaissance to the program in memory, he can identify which blocks the program has replaced with a dynamic `jmp` instruction that points to the block's new (changing) address. This means that the ROP payload can (1) load the target address of one of said `jmp` instructions, (2) use the target address as an offset for the desired attack gadet, and (3) compute the location of the desired attack gadget. However, this assumes that (1) the attacker already access to gadgets that can load these addresses and perform the required arithmetic, and that (2) all of the desired gadgets are protected by relocation and not by the other code randomization techniques used in our prototype. Alternatively, chronomorphic binaries could compute their jumps to relocated blocks using control-flow-sensitive values, so that relocated block addresses cannot be trivially looked up. The performance overhead of this approach might be reasonable, based on the density of relocated blocks, but this is an empirical question. Even in light of chronomorph-tailored ROP attacks, chronomorphic programs offer significantly more protection than the other techniques reviewed in Section II in a fully-observable setting on legacy hardware and software.

We do not presently protect the morph table, which resides outside of the binary. While chronomorphic binaries do not rely on obscurity for security, an attacker's chances of success would be higher if he has access to the morph table describing how the binary can change itself. Straightforward encryption techniques should allow us to protect the morph table.

We can potentially support multi-threading— and also ensure safe function continuation— by automatically injecting control flow monitors into diversifiable functions. Control

flow monitors, similar to those used for runtime-injected aspect-oriented programming [22], can be certified by well-established model-checking techniques [23]. These control flow monitors can determine during runtime when threads actually have an active stack frame for a given function, so runtime diversification will not modify those functions. These control flow monitors could also implement thread synchronization to prevent threads from entering a function that is currently being rewritten. Despite their certifiability by model-checking, these thread monitoring and synchronization techniques may incur high performance overhead, so this remains an open empirical question.

These challenges represent areas of future research and development for chronomorphic programs. Our prototype tool and preliminary analyses demonstrate that chronomorphic binaries reduce the predictability of code reuse attacks for single-threaded programs, and we believe that these avenues of future work will improve the safety and robustness of chronomorphic binaries in complex multi-threaded applications.

ACKNOWLEDGMENTS

This work was supported by The Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under contract FA8650-10-C-7087. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Approved for public release, distribution unlimited.

REFERENCES

[1] S. E. Friedman, D. J. Musliner, and P. K. Keller, "Chronomorphic programs: Using runtime diversity to prevent code reuse attacks," in Proceedings ICDS 2015: The 9th International Conference on Digital Society, Feb. 2015.

[2] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007, pp. 552–561.

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

[4] E. J. Schwartz, T. Avgerinos, and D. Brumley, "Q: Exploit hardening made easy." in USENIX Security Symposium, 2011, pp. 25–41.

[5] J. Salwan and A. Wirth, "Ropgadget," URL http://shell-storm.org/project/ROPgadget, 2011.

[6] A. Bittau, A. Belay, A. Mashtizadeh, D. Mazieres, and D. Boneh, "Hacking blind," in Proceedings of the 35th IEEE Symposium on Security and Privacy, 2014, pp. 227–242.

[7] J. Hiser, A. Nguyen-Tuong, M. Co, M. Hall, and J. W. Davidson, "Ilr: Where'd my gadgets go?" in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 571–585.

[8] S. E. Friedman, D. J. Musliner, and J. M. Rye, "Improving automated cybersecurity by generalizing faults and quantifying patch performance," International Journal on Advances in Security, vol. 7, no. 3–4, 2014, pp. 121–130.

[9] L. Davi, A.-R. Sadeghi, and M. Winandy, "Ropdefender: A detection tool to defend against return-oriented programming attacks," in Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM, 2011, pp. 40–51.

[10] H. Shacham et al., "On the effectiveness of address-space randomization," in Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004, pp. 298–307.

[11] M. Franz, "E unibus pluram: massive-scale software diversity as a defense mechanism," in Proceedings of the 2010 workshop on New security paradigms. ACM, 2010, pp. 7–16.

[12] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the gadgets: Hindering return-oriented programming using in-place code randomization," in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 601–615.

[13] R. Wartell, V. Mohan, K. W. Hamlen, and Z. Lin, "Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code," in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 157–168.

[14] A. Gupta, S. Kerr, M. S. Kirkpatrick, and E. Bertino, "Marlin: A fine grained randomization approach to defend against rop attacks," in Network and System Security. Springer, 2013, pp. 293–306.

[15] C. Giuffrida, A. Kuijsten, and A. S. Tanenbaum, "Enhanced operating system security through efficient and fine-grained address space randomization." in USENIX Security Symposium, 2012, pp. 475–490.

[16] M. Backes and S. Nürnberger, "Oxymoron: making fine-grained memory randomization practical by allowing code sharing," in Proceedings of the 23rd USENIX conference on Security Symposium. USENIX Association, 2014, pp. 433–447.

[17] S. Crane et al., "Readactor: Practical code randomization resilient to memory disclosure," 2015.

[18] L. Davi, C. Liebchen, A.-R. Sadeghi, K. Z. Snow, and F. Monrose, "Isomeron: Code randomization resilient to (just-in-time) return-oriented programming," Proc. 22nd Network and Distributed Systems Security Sym.(NDSS), 2015.

[19] S. Crane, P. Larsen, S. Brunthaler, and M. Franz, "Booby trapping software," in Proceedings of the 2013 workshop on New security paradigms workshop. ACM, 2013, pp. 95–106.

[20] R. El-Khalil and A. D. Keromytis, "Hydan: Hiding information in program binaries," in Information and Communications Security. Springer, 2004, pp. 187–199.

[21] D. Babić, L. Martignoni, S. McCamant, and D. Song, "Statically-directed dynamic automated test generation," in Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), Toronto, ON, Canada, Jul. 2011.

[22] K. W. Hamlen and M. Jones, "Aspect-oriented in-lined reference monitors," in Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security. ACM, 2008, pp. 11–20.

[23] M. Sridhar and K. W. Hamlen, "Model-checking in-lined reference monitors," in Verification, Model Checking, and Abstract Interpretation. Springer, 2010, pp. 312–327.

Building Trusted and Real Time ARM Guests Execution Environments for Mixed Criticality

With T-KVM, a hypervisor architecture that implements an hardware isolated secure and real time environment

Michele Paolino, Kevin Chappuis, Alvise Rigo, Alexander Spyridakis, Jérémy Fanguède,
Petar Lalov and Daniel Raho
Virtual Open Systems
Grenoble, France
Email: {m.paolino, k.chappuis, a.rigo, a.spyridakis, j.fanguede, p.lalov, s.raho} @virtualopensystems.com

***Abstract*—The new ARMv8 architecture is targeting the server, Network Functions Virtualization (NFV), Mobile Edge Computing (MEC) and In-Vehicle Infotainment (IVI) market segments. At the same time, it will empower Internet of Things (IoT), Cyber Physical Systems (CPS), automotive Electronic Control Units (ECU), avionics and mixed criticality devices. In this context, virtualization is a key feature to enable the cloud delivery model, to implement multitenancy, to isolate different execution environments and to improve hardware/software standardization and consolidation. Since guaranteeing a strict isolation of both the data and the code executed in Virtual Machines (VMs) counts today more than ever, the security of the hypervisor and its guests has become dramatically important. This paper extends Trusted Kernel-based Virtual Machine (T-KVM) [1], an architecture for the KVM-on-ARM hypervisor proposed to satisfy the above market trends, in the direction of an efficient and high performance interrupt management. T-KVM integrates software/hardware components to isolate guest Operating Systems (OSes) and enable Trusted Computing along with mixed criticality in ARM virtual machines. It combines four isolation layers: ARM Virtualization and Security Extensions (also known as ARM VE and TrustZone), GlobalPlatform Trusted Execution Environment (TEE) APIs and SELinux Mandatory Access Control (MAC) security policy. In this paper, the T-KVM architecture and its interrupt management features are described in detail, as well as its key implementation challenges and system security considerations. Lastly, a performance evaluation of the proposed solution is presented.**

Keywords–Trusted KVM; ARMv8 Trusted Computing; ARM Virtualization; Mixed Criticality; Real Time.

I. INTRODUCTION

The use of virtualization in ARM platforms is rapidly increasing due to the deployment of SoCs based on this architecture in different environments such as: servers, Cloud and High Performance Computing (HPC), NFV, MEC, IoT, CPS, smart devices, avionics, automotive, etc.

Virtualization enables multiple OSes to run unmodified on the same hardware, thus sharing system's resources such as memory, CPUs, disks and other devices. These resources are frequently target of specific virtualized environment attacks (e.g., CPU cache [2], memory bus [3] and VM's devices [4] [5]). For this reason, the security of the virtualized systems is critical. Historically, isolation has been used to enhance the security of these systems [6], because it reduces the propagation risks in compromised environments [7].

T-KVM [1] is a novel security architecture for virtualized systems, which adds three isolation layers (ARM TrustZone, GlobalPlatform TEE API and SELinux) on top of the standard VMs isolation provided by the hypervisor and provides support for the concurrent execution of a hypervisor and a real time operating system. The main contribution of this paper is the analysis and benchmark of T-KVM in respect to the interrupt management mechanisms during the concurrent execution of two operating systems into the TrustZone execution worlds. In fact, T-KVM targets to provide a fast and efficient interrupt handler, which allows to meet real-time constraints while providing high performance for the guests applications. For this reason, the ARMv8 interrupt management basics and the possible T-KVM implementations have been described and analyzed in this paper, aiming to compare different interrupt management approaches in order to select the best solution for mixed criticality systems.

As mentioned, T-KVM proposes four isolation layers: KVM, ARM TrustZone, GlobalPlatform TEE and SELinux. The former is considered the most popular hypervisor deployed in OpenStack [8], which is a key solution for Cloud, NFV and HPC computing. KVM for ARM is part of the Linux kernel starting from the version 3.9; it is the Linux component that, exploiting the ARM Virtualization Extension, allows to create a fully-featured virtualization environment providing hardware isolation for CPU, memory, interrupts and timers [9]. TrustZone is an hardware security extension for ARM processors and Advanced Microcontroller Bus Architecture (AMBA) devices [10] designed to drastically improve security inside the ARM ecosystem. The extension starts from the assumption that a system, in order to deliver secure services, has to decouple the resources used for general purpose applications from those that handle security assets. To this end, TrustZone creates two hardware isolated partitions in the system: the Secure and the Non Secure World. While the Non Secure World runs a standard OS with optionally a hypervisor, the Secure World contains, handles and protects all the sensitive data (credit card

information, customers list, passwords, etc.) and code (e.g., real time tasks, ciphers functions, etc.) of the system. These two worlds are linked together through the GlobalPlatform TEE API [11] [12], a set of specifications for a secure Remote Procedure Call (RPC) mechanism between the trusted and non-trusted compartment of the system. At the time of writing, these specifications do not support virtualization, preventing the use of Trusted Platform Module (TPM) services inside virtual machines. This work addresses this limitation proposing a design of a set of virtualization extensions to enable the guest operating systems to make use of TPM services provided by the TrustZone Secure World. The latter T-KVM isolation layer is Security-Enhanced Linux (SELinux), a Mandatory Access Control (MAC) solution, which brings type enforcement, role-based access control and Multi-Level Security (MLS) to the Linux kernel [13]. By means of these, SELinux confines processes in security domains, where the interaction with other processes and files is permitted only if there is a specific SELinux policy rule that allows it.

In T-KVM, the above technologies are combined and adapted to work together, providing high security for guest applications and strong isolation for the Secure World OS/Real-Time OS (RTOS), without the need of specific hardware or software. As a matter of fact, the proposed architecture relies on open source (KVM and SELinux) components, public specifications (GlobalPlatform TEE Internal and Client APIs) and available hardware features (ARM TrustZone and VE). For these reasons, T-KVM can be easily ported to currently available ARM platforms, such as Cloud Infrastructure systems based on OpenStack, automotive ECU, avionics platform and other embedded systems.

The T-KVM architecture matches perfectly with server platforms and user devices but also with mixed criticality systems, where different levels of criticality need to interact and co-exist on a common hardware platform (e.g., infotainment and instrument clusters in avionics, or Advanced Driver Assistance Systems - ADAS - and web browsing in automotive, etc.), because it makes possible to run in the Secure World a critical security sensitive application (e.g., encryption algorithms, real time tasks, etc.) totally isolated from the Normal world.

The remaining part of this paper is organized as follows: Section II introduces the main security components of the proposed architecture. Section III contains details about the T-KVM architecture, its implementation and security considerations. Section IV describes the T-KVM interrupt management for real time and mixed criticality systems while Section V presents a performance analysis of the overhead and interrupt latencies of the proposed solution. The related work is presented in Section VI and Section VII concludes the paper.

II. THE SECURITY COMPONENTS

In this section, the isolation layers, which characterize the T-KVM architecture, are described.

A. KVM hypervisor

A hypervisor is a software layer, which is able to create virtual instances of hardware resources such as CPUs, memory, devices, etc. in order to enable the execution of multiple operating systems on the same hardware. Different implementation approaches lead to different hypervisor types: a type 1 hypervisor, is a bare metal hypervisor, which runs directly on the hardware (XEN or VMWare ESX). A type 2 hypervisor is, on the other hand, a hypervisor, which runs inside an operating system (Oracle VirtualBox or VMWare Workstation) at the application layer. Usually, the latter is used in less critical applications [14] because of its dependency from the underlying operating system.

KVM is a hypervisor included in the Linux kernel and available for ARM, x86 and s390 architectures. It is neither a type 2 hypervisor because it does not run as a normal program inside Linux, nor is a typical type 1 hypervisor, because it relies on the Linux kernel infrastructure to run. KVM exploits the CPU Virtualization Extensions to execute guest's instructions directly on the host processor and to provide VMs with an execution environment almost identical to the real hardware. Each guest is run in a different instance of this execution environment, thus isolating the guest operating system. For this reason, this isolation has been used for security purposes [15] [16] [17] [18] in many scientific works. In the ARM architecture, the KVM isolation involves CPU, Memory, Interrupts and timers [9].

B. TrustZone

ARM TrustZone is a set of hardware security extensions for ARM processors and AMBA devices. With TrustZone, the hardware platform is split in two parts, the Secure and the Non Secure Worlds. In order to isolate these two compartments, TrustZone requires: CPU with ARM Security Extensions (SE) along with TrustZone compliant Memory Management Unit (MMU), AMBA system bus, interrupt and cache controllers. Hence, the isolation provided by TrustZone includes CPU, AMBA devices, interrupts, memory and caches.

The Secure World is considered trusted, and is responsible for the boot and the configuration of the entire system. In fact, the CPU has banked registers for each World, and security specific configurations can be performed in Secure World mode only. This compartment contains the root of trust of the system and protects sensitive data. The access to AMBA peripherals such as fingerprint readers, cryptographic engines, etc. can be restricted only to the Secure World, thus protecting security devices.

On the other hand, the Non Secure World is intended to be the user's World. In this untrusted compartment, a standard operating system (i.e., Android or Linux) is run. Security sensitive operations such as the access to a secret or the interaction with a real time task are provided to the user's application running in this compartment by the services run in the Secure World.

These two compartments interact with each other through a specific CPU mode, namely the Monitor Mode. It typically runs a secure context switch routine and is capable of routing interrupts, depending on the configuration, either to the Secure or Non Secure World.

Moreover, the use of the ARM VE Extensions, and consequently of KVM, is possible only in the Non Secure World.

ARM TrustZone is compliant with the GlobalPlatform TEE System Architecture specification [19], which defines the attributes that the hardware must have to properly execute a TEE.

C. GlobalPlatform TEE

GlobalPlatform specification defines the TEE as an execution environment, which provides security features such as isolated execution, integrity of Trusted Applications (i.e., applications run in the TEE) along with confidentiality of their assets [19]. This is done by means of specific hardware capabilities of the system, such as ARM TrustZone. The TEE protects Trusted Applications (TA) and their data from the Rich Execution Environment (REE), the environment where a standard operating system such as Linux or Android is run. Figure 1 depicts the standard architecture of a GlobalPlatform TEE compliant system.

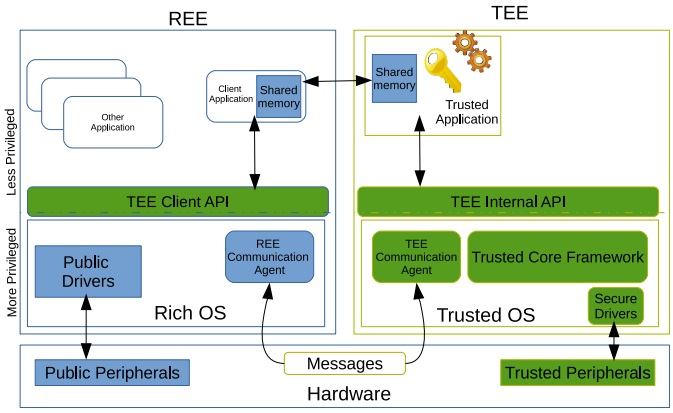


Figure 1. Standard TEE architecture

In order to isolate the TEE from the REE, GlobalPlatform provides a set of specifications, which include the following software components [11] [12]:

- The Trusted Core Framework is a common abstraction layer, which provides to the TEE Internal API OS-like functions, such as memory management, entry points for TAs, panic and cancellation handling, TA properties access, etc.
- The TEE Client API is an Inter Process Communication (IPC) API that deals with the communication between the REE and the TEE. It allows the applications in the REE (Client Applications or CAs) to leverage the services offered by the TEE.
- The (TEE and REE) communication agents provide support for messaging between the CAs and the TEE. They interact with the Monitor mode to request a context switch between the two Worlds.
- The TEE Internal API allows the TAs to leverage the services offered by the TEE through the following APIs: Trusted Storage for Data and Keys, TEE Cryptographic Operations, Time, and TEE Arithmetical.

Lastly, it is worth to mention that the deployment of a GlobalPlatform compliant solution enables the use of existing TA and CA applications. This a is very important factor, especially in an environment such as the embedded trusted computing, where by tradition the security solutions were developed each time from scratch to address new device families.

D. SELinux

SELinux is a software implementation of the MAC security policy available in the Linux kernel as Linux Security Module (LSM). The key feature of MAC is that the access control decisions are not at discretion of individual users, root included [13]. Thus, once the system security policies have been defined and loaded at boot time in the kernel, they can not be modified. In this way, the subject (e.g., a process) access to objects (e.g., file, socket, etc.) is enforced in the system.

The very same concept can be applied to virtual machines using sVirt, which is a feature of the libvirt library. sVirt installs a set of virtualization specific security policies and automatically tags VMs and their resources in order to isolate guest systems. This isolation prevent any access to VM's resources (disk/kernel files, shared memory, etc.) from external subjects (other VMs, the root user, etc.).

For this and for performance reasons [20], the use of SELinux in virtualized systems is encouraged.

III. THE TRUSTED HYPERVISOR: T-KVM

T-KVM is a secure hypervisor architecture based on KVM, which combines a Trusted Computing solution such as TrustZone with GlobalPlatform TEE and SELinux, to enable both Trusted Computing and RT support for ARM based platforms. In Figure 2, all the components described in Section II are shown together, composing the T-KVM architecture.

In T-KVM, the GlobalPlatform TEE and REE are respectively implemented inside the TrustZone Secure and Non Secure Worlds. For this reason in the remaining part of this paper, Secure World/TEE and Non Secure World/REE are used as synonyms. These two hardware-isolated environments are linked together with a virtualization-enabled implementation of the GlobalPlatform TEE specifications. The virtualization provided by KVM further isolates the user's applications, enabling the use of different operating systems. This eases multitenancy in server and Cloud environments, and enables BYOD paradigm in smart devices. In addition, SELinux isolates in software the virtual machines, protecting guests and

their resources from the other virtual machines and the host itself (e.g., malicious cloud administrators, host privilege escalation exploits, etc.). Lastly libvirt, the main virtualization API used by OpenStack to interact with KVM, takes automatically care of the policy configuration and the tag assignment through its component sVirt.

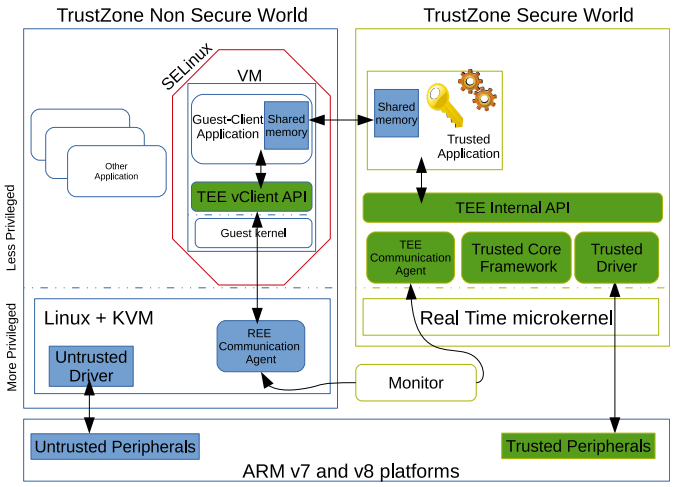


Figure 2. T-KVM architecture, which includes KVM, TrustZone, SELinux and virtualized TEE

A. *Implementation details*

The following part of this manuscript lists the T-KVM implementation challenges and proposes viable solutions.

1) Trusted boot: The first step of the system's chain of trust is performed during the boot procedure. In fact, when the machine boots, the security configuration of the system is not yet in place, and as a result the system is vulnerable to attacks, which target to replace the boot procedure.

In order to minimize this risk, the T-KVM's first stage bootloader is a tiny program stored in a on-chip Read Only Memory (ROM) along with the public key needed for the attestation of the second stage bootloader. Since the first stage bootloader is stored in a read-only memory, it can not be updated and, therefore, it is critical for the security of the system. Soon after the initialization of the key system components, it checks the integrity and boots the second stage bootloader, which is located in an external non-volatile memory (e.g., flash memory). The second stage bootloader then loads the real time microkernel binary in the system's Secure World memory and boots it.

The third stage bootloader of the T-KVM Trusted boot mechanism is a Trusted Application inside the TEE. In fact, when the Secure World OS is up and running, a specific TA checks the integrity of the Non Secure World OS binary (i.e., the Linux kernel) and its bootloader (fourth stage). If this last security check is successful, the fourth stage bootloader runs the Non Secure OS and the system can be considered running. On the other hand, if only one of these checks fails, the boot process will be stopped and the machine will enter in a secure and not operational state. Figure 3 shows the T-KVM Trusted boot chain of Trust.

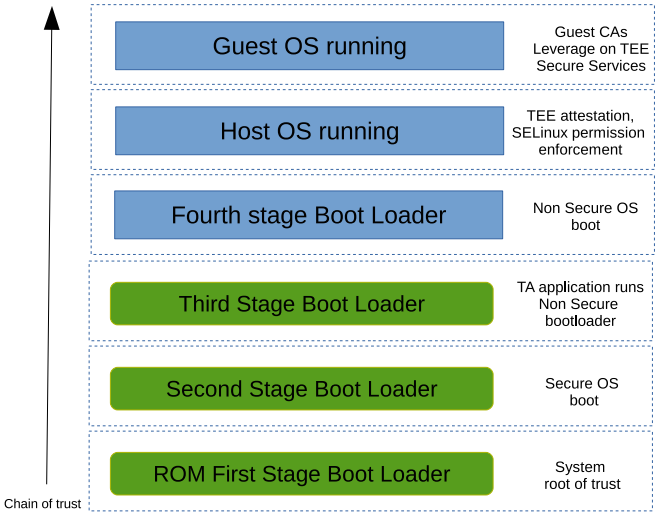


Figure 3. T-KVM Trusted boot procedure

The Trusted boot process is the key element for the attestation of the user space applications because it ensures the integrity of the chain of trust. T-KVM runs in the TEE an attestation service, which is able to check at any moment the integrity of its key components i.e., QEMU, libvirt, the VMs and their resources, etc. libvirt in particular, is extended to attest the VMs identity and integrity in an event-driven manner (e.g., a new VM is booted/shut down, a new device is plugged into the guest, etc.), assuring to users and cloud administrators/providers the authenticity of the workloads run on the hardware. The security assets (fingerprints, keys, etc.) of these binaries are stored in the Secure World and by consequence can be easily updated when necessary.

2) GlobalPlatform TEE support for virtualization: The main novelty introduced by T-KVM is the support for Trusted Computing inside the Virtual Machines. The virtualization of the TEE functions is of utmost importance for the T-KVM architecture, because it links together the applications run in the VMs with the secure/RT services available in the TrustZone Secure World. At the time of writing, the use of the TEE functions in guest operating systems is not included in the GlobalPlatform API Specification. To enable this feature, T-KVM virtualizes the GlobalPlatform TEE APIs, executing the TEE Client API directly in the Guest Operating System. In order to be as much as possible compliant with the GlobalPlatform Specification and to be able to run CAs also at the host level, T-KVM TEE Client API is the only virtualization aware component.

This awareness needs support from the hypervisor infrastructure, for this reason, as depicted in Figure 4, a specific QEMU device is used to implement the TEE control plane and

set up its data plane. All the requests (e.g., initialization/close session, invoke command, etc.) and notification of response are sent to the TEE Device, which delivers them either to the TAs or to the CAs running on the guest OS. To provide good data throughput and latency performance, the data plane is based on a shared memory mechanism. Thus, when a response notification arrives from the TrustZone Secure World, the TEE device notifies with an interrupt its driver, which forwards the related information to the Guest-Client Application. The Guest-CA is now able to read the data from the shared memory, without involving the TEE device in the data transfer.

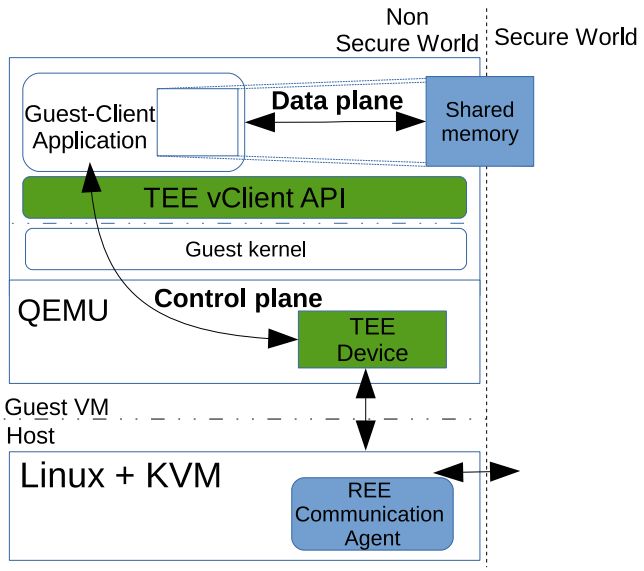


Figure 4. TEE support for Virtual Machines in T-KVM

3) Shared memory: T-KVM needs a zero copy shared memory mechanism to share data between the two TrustZone Worlds and between the virtual machine and the host. The latter in particular is very important in systems where VMs need to communicate with each other frequently (e.g., NFV, HPC, MEC, etc.). Host-guest only shared memory mechanisms which provide high performance and low latency already exist for the KVM hypervisor [21] [22]. What these mechanisms actually lack is the support for TrustZone and, therefore, they are not able to support communication between the TrustZone Non Secure and Secure worlds.

By design, the TrustZone Secure World is able to access the full Non Secure World address space. For this reason, the Trusted Applications are not able to read/write the content of the VMs shared memory unless they know the address where the shared memory area begins. In order to pass this information, the TEE device control plane extends the T-KVM shared memory mechanism, enabling it to send the shared memory address to the Secure World applications. This mechanism need to be secured, especially in the Non Secure World, to prevent attacks and information leakage. T-KVM relies on SELinux to define specific access rules for shared memory and to enforce the shared memory access only to the interested parties.

The encryption of the shared memory area is consciously not considered because, unless hardware accelerators are present in the platform, there would be a performance loss.

4) Secure World: One of the most important parts of the T-KVM architecture is the software running in the TrustZone Secure World. The operating system running in the Secure World should be fast, secure, real-time and free of programming errors (implementation correctness).

The T-KVM architecture empowers the Secure World environment with a real time microkernel. The primary motivation behind microkernels is the small code footprint, which lead to a smaller attack surface and an easier process of formal verification of the code. Moreover, thanks to their real time capability, they enable the deployment of T-KVM in mixed criticality environments such as avionics and automotive. In this context, good candidates are for example seL4, an open-source third-generation microkernel based on L4 and formally verified for functional correctness [23], or FreeRTOS [24] one of the most popular real time operating system for embedded systems.

The microkernel will run in its user space the implementation of the GlobalPlatform APIs, the secure device drivers and the TAs. In order to do this, the Secure World OS does not use the main platform storage device to store files and the security assets of the system. An external, non-volatile memory configured by the Secure World as not accessible by the Non Secure World, is used to this purpose.

Finally, a possible alternative to microkernels is a secure library running in the Secure World such as OPTEE [25]. Despite this solution has a code footprint even smaller than a microkernel, T-KVM uses a microkernel because of its real time features and a higher flexibility for TA developers.

B. Security considerations

Virtual Machines are widely used because of their flexibility and capability to run any operating system. Nonetheless, in order to achieve a higher security level of the system, it is suggested to run single-application operating systems in the T-KVM virtual machines. An example of such an operating system is OSv [26], an open source solution, which is going to be ported to the ARM v8 architecture. For this reason, this work does not discuss the security of the applications inside the virtual machines.

The threat model considered in this paper allows the attacker to completely control one or more virtual machines, both at user and kernel space level. In addition, the cloud administrator, who is permitted to remotely control the virtualized system, is considered as a potential attacker for the identity and integrity of the data.

The security of T-KVM is based on two main assumptions: the attacker does not have physical access to the virtualized system and the first stage bootloader is flawless (thus, the chain of trust is not compromised).

T-KVM has been designed to be compliant with additional hardware accelerators and security modules. For example, SecBus [27] can be used to protect the system against physical attacks on the memory components (e.g., cold boot), Direct Memory Access (DMA) attacks and on-board probing of the external memory bus. Solutions like Network-on-Chip (NoC) Firewall [28], can enhance the compartment isolation granularity at VM level, protecting the system against logical attacks (virus, Trojans) or security vulnerabilities, e.g., corrupted DMA engines.

IV. The T-KVM interrupt management

This section describes how the proposed architecture aims to minimize the interrupt latencies in order to meet real time constraints by providing fast responses.

A. T-KVM interrupt allocation

ARM v7, v8 and earlier architectures have been designed to support two interrupt types: the Fast Interrupt Request (FIQ) for low latency interrupt handling, and the more general Interrupt Request (IRQ), which is commonly available also in other architectures.

The former has higher priority (IRQs are automatically masked by the CPU core when an FIQ arrives) and can directly use some banked registers without the overhead of saving/restoring them through push/pop instructions.

T-KVM takes advantage of the ARM architecture interrupt management design [29], by programming the Secure world to respond only to FIQs and the Normal world to handle IRQs only. This allows critical applications to benefit from fast and high priority interrupts, while isolating them from the Non Secure IRQs.

Lastly, is to be said that the ARM v8 architecture adds the possibility to assign interrupts exclusively to the Monitor (ARM v8 Exception Layer 3). This is an interesting feature for solutions like T-KVM, because it enables the system to isolate interrupts addressed to the Secure World from interrupts addressed to the TrustZone monitor. However, currently available platforms equipped with the Generic Interrupt Controller version 2 (GICv2) such as the ARM JUNO board do not support this feature. T-KVM has been designed to run on existing platforms, and will be adapted and extended when platforms equipped with GICv3 [30] will be available.

The following subsections describe the basic interrupt handling mechanisms for each type of the world execution.

B. Secure World interrupt handling

During the execution of the Secure World, IRQs and FIQs are enabled. FIQs interrupts are handled by the RTOS (red arrow in Figure 5), while IRQs are redirected to the Monitor (green arrow in Figure 5). This minimizes the FIQ interrupts latency during the Secure World execution, because these are directly caught in the handler of the critical application, avoiding any context switch overhead. On the other hand, if an IRQ occurs during the Secure World execution, the Secure OS is preempted and the IRQ is caught in the monitor. At this point, in order to handle the IRQ to the Normal World, the monitor will provoke a context switch by saving the Secure World context and restoring the Normal World, before executing its IRQ handler.

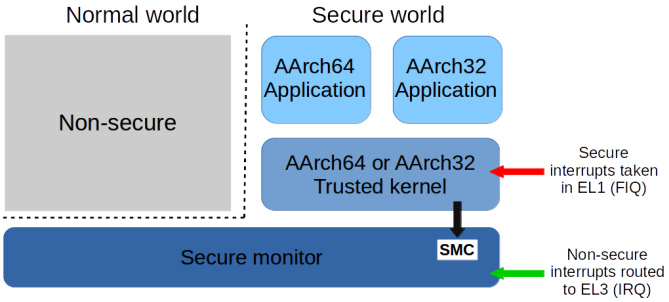


Figure 5. Exception interaction during the Secure world execution

This interrupt management allows IRQs to preempt the Secure world execution when the critical application does not execute a sensitive part (e.g., when it is executing a FIQ handler routine). However, in some specific cases, IRQs can be disabled to prevent the Normal World from interrupting the execution of the Secure World. In that case, the Normal world executes only when there is an explicit request from the Secure world. This is achieved through the Secure Monitor Call instruction.

C. Normal World interrupt handling

Similarly to what happens in the Secure World, both FIQs and IRQs are enabled during the execution of the Normal World: IRQs are directly handled in the Normal World (green arrow in Figure 6) while FIQs are redirected to the TrustZone monitor (red arrow in Figure 6). When a FIQ occurs, the Non Secure OS is immediately preempted in order to propagate the FIQ to the Secure World as soon as possible. This lowers latencies and helps the critical application to meet real time constraints. At this point, the FIQ is caught in the monitor, which executes a world context switch, by saving the Normal World context and restoring the Secure World one, in order to handle the FIQ.

During the FIQ handling into the monitor layer the FIQ mask is enabled and prevents any preemption by another FIQs having higher priority. Then, once the execution control is given to the Secure world, the FIQ management is handled by the trusted application. Therefore, the Secure world could decide to disable the FIQ mask when the critical part is over.

So, once the control is given to the Secure world, the FIQ management is overseen by the FIQ handler of the secure application. Therefore, the Secure world could decide to disable the FIQ mask when the critical part of its FIQ handler is over.

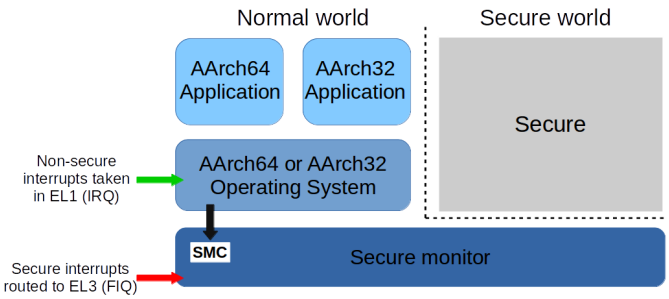


Figure 6. Exception interaction during the Normal world execution

V. Experimental Results

In this section, the result of experimental tests performed on a T-KVM prototype are shown and analyzed. These performance measurements have been performed on an ARMv8 Juno board, which is equipped with two Cortex-A57 and four Cortex-A53 in big.LITTLE configuration. All the workloads for the performance analysis have been executed on the Cortex-A53, which is the default CPU that the platform uses to execute the Secure World OS.

A. SMC propagation

The different isolation layers, which compose T-KVM provide high security, but at a cost of additional overhead. As a matter of fact, a request for a security service from a virtual machine has to pass through the TEE, the host system and SELinux to arrive in the TrustZone Secure World.

For the T-KVM performance analysis of this paper, we focused on the hardware isolation provided by the hypervisor and TrustZone, as the SELinux performance has been measured by the authors in the past [20], and the TEE overhead will be detailed in future works.

For this reason, following the path of a Secure Monitor Call (SMC) from the guest to the Secure World (and then back in the guest) we measured the overhead introduced by T-KVM.

SMC has been added to the ARM instruction set by the ARM Security Extension and it is used to request the execution of a software routine in the TrustZone Secure World passing through the TrustZone Monitor. In the standard KVM implementation, when the SMC instruction is run by a guest OS, its execution is trapped by KVM, which injects an undefined instruction in the guest, forcing it to handle this accordingly. In T-KVM instead, when such instruction is run, the hypervisor traps the guest SMC execution, modifies its arguments and forwards them to the TrustZone Secure World.

In this scenario, two SMC context switches are involved: firstly from Guest to Host, then from Non Secure to Secure World Mode. The overhead assessment of these two context switch operations is the target of the following analysis.

For the first measurement, we implement a bare metal binary blob for KVM, which initializes the Performance Monitoring Unit (PMU) and executes the SMC instruction in the guest. The SMC is then trapped by KVM, which has been modified to immediately return the control to the VM. As soon as the program flow returns back to the guest, it checks the PMU cycle counter status and calculates the overhead. In this way, we are able to measure the overhead of a round-trip context switch between the Guest and the Host when an SMC call is executed.

On the other hand, for the measurements of the context switch overhead between the Non Secure and the Secure Worlds, the PMU cycle counter is set by a Linux kernel module in the host, which executes soon after the world switch request (i.e., the SMC instruction). This provokes an additional switch to the TrustZone Monitor, which saves the Normal World registers, loads the Secure World status and finally jumps to the Secure World. In order to measure the T-KVM context switch cost and not add further overhead, the Secure World immediately runs the SMC call, without executing any meaningful operation. When this instruction is executed in the Secure World, it provokes again a switch to the Monitor Mode, which will now save the secure world context, restore the non secure context, and jump back to the Non Secure World. As soon as the context switch is completed, the Linux kernel reads the PMU cycle counter state and computes the overhead.

The results of both of the above measurements are defined minimal because they are not considering any additional work performed at the destination where they are trapping to. Moreover, in both cases the system frequency is fixed at boot time. However, it is to be considered that, in a real scenario, a trap is followed by the execution of emulation code for a Guest-Host trap, or some secure service for a Non Secure-Secure trap.

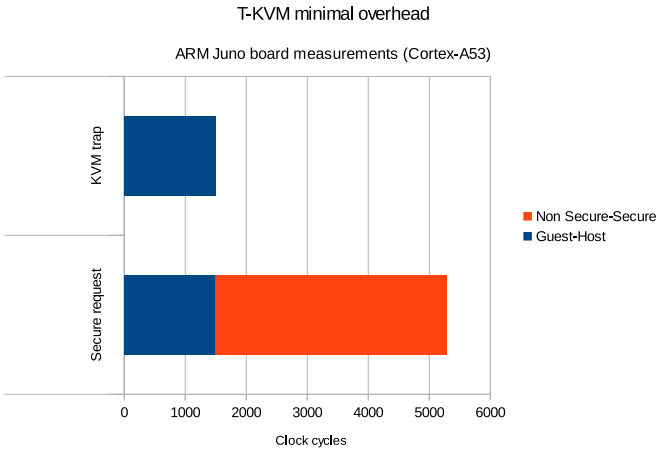


Figure 7. T-KVM overhead measurements

Each test has been repeated five hundred times, running Linux version 3.17 in both the host and the guest environments. ARM Trusted Firmware [31] has been used as firmware infrastructure.

Figure 7 shows that for a Secure request, which is the sum

of the two measurements described before in this section, the cost is in average 5200 clock cycles. This value represents the minimum overhead that a guest system has to pay to request a secure service to the T-KVM Secure World (the Guest-Host minimum overhead is about 1400, while the Non Secure-Secure is about 3700). This has been compared with the minimum overhead that KVM spends to trap the SMC instruction and perform a context switch between the guest and the host, which is what has been described above in the Guest-Host context switch measurement description. This result (in average about 1400 clock cycles) has been measured with the SMC instruction, but it is valid for all the instructions trapped in KVM, as we did not add any specific code to trap the SMC instruction to the standard KVM implementation.

Finally, it is important to notice that the Non Secure-Secure context switch is 2.5 times slower than the Guest-Host. The main reason for this behaviour is in the number of instructions needed to complete the two operations. In particular, the number of registers that the system has to save and restore for the Guest-Host context switch is significantly lower.

B. FIQ latency

As described in the Section III, the T-KVM architecture isolates the time critical applications in the TrustZone Secure World. For this reason, the following performance analysis is focused mainly on the interrupt latencies at the Secure World side, where a RTOS is running.

The proposed architecture, uses FIQ interrupts only for the RTOS, for the reason explained in Section IV. However, the allocation of Fast Interrupts to the RTOS can be implemented in different ways. Hence, following the FIQ path from the vector table to the end of FIQ handler in the Secure world, the overhead introduced by the T-KVM interrupt management has been compared with the performance of different possible implementations:

- **RTOS only** This case represents the T-KVM interrupt management described in Section IV, where FIQs interrupts are handled by the RTOS, while IRQs are redirected to the Monitor.
- **Monitor/RTOS** As above, FIQs are caught in the Monitor. However, in this case, the monitor does not acknowledge the interrupt if the FIQ is not dedicated to the monitor. It forwards FIQs directly to the Secure world, before disabling FIQ trap in the monitor layer (EL3). Therefore, if the FIQ is redirected to the Secure world, the FreeRTOS FIQ handler has to generate an additional SMC at the end of FIQ process in order to return in the monitor to re-enable FIQ trap.
- **Monitor only** The TrustZone monitor (EL3) has full control of the FIQs. FIQs are always trapped in the monitor vector, which can read/write GIC registers to manage the FIQ (activation, acknowledgement, etc.). If the FIQs is not for the Monitor itself, it propagates the FIQ to the Secure world (FreeRTOS).

For all the different FIQ paths described above, a system timer has been programmed to generate periodic interrupts used by the FreeRTOS Tick management to schedule different real-time tasks. Moreover, the FIQ handler installed on the FreeRTOS side, is the same for all tests.

Two different test cases will be taken in consideration for each possible implementation. The first is the measurement of the FIQ latency when the interrupt occurs during the execution of the Normal World (and thus requires a world context switch), and the second is the measurement of the FIQ process time when it occurs during the execution of the Secure world. Moreover, to avoid any cache impact on the results, instruction and data cache memories have been disabled for all the tests.

Moreover, the measurements are split in three parts, each one representing the state of the time at a specific phase:

- ”FIQ trigger” is the time to trig the FIQ in the vector table when the timer reached a specified value.
- ”FIQ propagation in the monitor” is the time for the monitor to execute some operations to manage the FIQ before to redirect it to the Secure world.
- ”FreeRTOS FIQ processing” is the time execution of FIQ handler in FreeRTOS to process the Tick management.

The Timer is set in free-running mode with a frequency of 50 MHz.

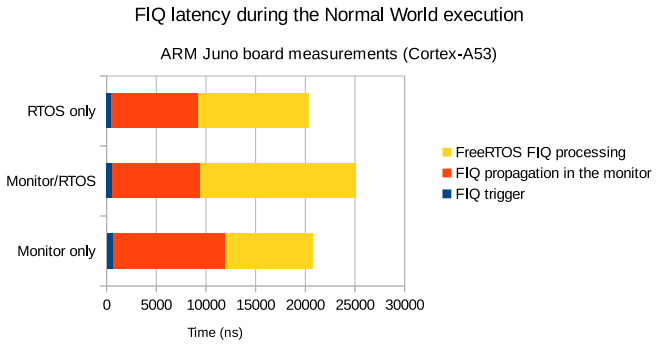


Figure 8. FIQ latency measurements during the Normal world execution

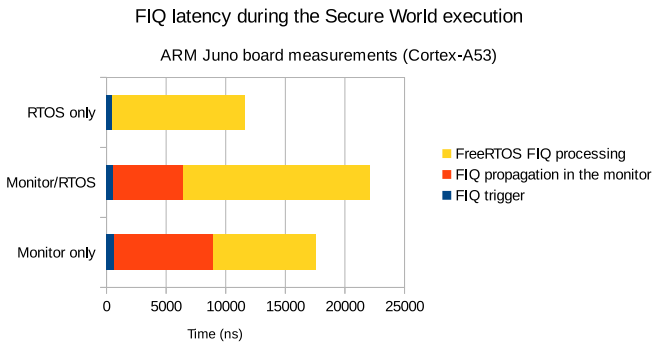


Figure 9. FIQ latency measurements during the Secure world execution

Each test has been repeated one hundred times, running Linux/KVM version 4.0 in the Normal World and a FreeRTOS v8.2.1 in the Secure World. The code needed to run FreeRTOS on the ARM Juno board has been shared by Virtual Open Systems with the FreeRTOS community [32]. Moreover, a modified version of ARM Trusted Firmware [31] developed by Virtual Open Systems, also publicly available as open source software [33], has been used as firmware infrastructure for the monitor layer.

Figure 8 and Figure 9 show that all ”FIQ trigger” parts have approximately the same value for all different tests. This is expected because it corresponds to the hardware time needed to generate the FIQ, and as a consequence it not influenced by the chosen software implementation. Moreover, it is important to notice that the timer value measured to determine the ”FIQ trigger” time has been taken after the execution of few instructions in the vector table handler. So, the value is slightly larger that expected.

About the result of different tests during the execution of the Normal world, Figure 8 shows that the software solution implemented in the Monitor only case has no impact on the total FIQ handling time compared to the basic implementation in the RTOS only. However, the ”FIQ propagation in the monitor” time is more important in the Monitor only case because the monitor executes some instructions, which are normally executed in the FreeRTOS FIQ handler. For the Monitor/RTOS test, there is no overhead during the ”FIQ propagation in the monitor” part because, as the FIQ is dedicated to the Secure world, it forwards directly the FIQ to the FreeRTOS. In this case, an additional communication is implemented to re-enable FIQ trap in EL3 at the end of FreeRTOS FIQ handler and Figure 8 shows that this solution adds an important overhead in the ”FreeRTOS FIQ processing” part.

Additionally, in the case where the FIQ occurs during the execution of the Secure world, Figure 9 shows that the ”FIQ propagation in the monitor” time is lower than the same test realized during the Normal execution. This is expected because, as the world interrupted is the Secure and the monitor redirects the FIQ to the Secure world, the monitor does not execute the context switching operation in this case. Moreover, in the basic interrupt management of the RTOS only test, there is no monitor execution overhead because the FIQ is directly caught in the FreeRTOS FIQ handler.

VI. Related Work

T-KVM is a hypervisor architecture, which mainly targets security and isolation to run virtual machines and real time tasks together on the same hardware platform. Hypervisors' security is a controversial topic in literature. As a matter of fact, solutions like NoHype [34] [35] propose to secure the virtual machines removing the hypervisor, while others use the hypervisor isolation for security applications [16] [18]. In other scientific works, when compared with TrustZone as a security solution, the hypervisor proves a better flexibility, e.g., the Secure World is not able to interpose on and monitor all the important events in the Non-Secure World [17]. The proposed architecture considers the hypervisor as an additional isolation layer, while protecting the security assets through the ARM Security Extensions (TrustZone). T-KVM, combining both solutions and relying on the attestation enabled by the secure boot's chain of trust, is able to provide monitoring features and high security.

In fact, attestation and integrity checks are of paramount importance for the security systems because they allow system designers and administrators to consider a software component as trusted. SecVisor [36], HyperSentry [37] and SPROBES [38] propose different solutions designed for this purpose: the first checks the integrity of commodity OS kernels running the attestation code in hypervisor mode, thus not addressing virtualization. The second enables integrity measurement of a running hypervisor (i.e., XEN) through Intel TxT, hence targeting the x86 architecture. The latter uses TrustZone to enforce kernel code integrity, but without mentioning the attestation challenges in virtualized systems.

These systems are explicitly addressed by solutions such as vTPM [39] or sHype [40], both focusing their efforts on Intel architectures. vTPM proposes a mechanism for the virtualization of TPM functions, which dedicates a VM to route and manage the TPM requests, while sHype integrates the MAC security policy directly inside a typical type 1 hypervisor (i.e., XEN).

On the other hand, the solutions proposed by Narari [41] and Lengyel [42] are designed for ARM devices with virtualization extensions. The first proposes a security architecture with TrustZone, SELinux and virtualization, targeting resource constrained devices. The second combines a hypervisor (XEN) and MAC Security policies (XEN Security Modules), targeting high isolation between VMs but without mentioning TPM access for guest OSes. Both proposals lack of a solution to standardize the access to TPM functions such as the TEE.

VII. Conclusion and Outlook

This paper proposes T-KVM, a new security architecture for ARM v7 and v8 virtualized systems, providing architecture details and a performance analysis. T-KVM's architecture offers strong isolation for guest applications by means of the KVM

hypervisor, ARM TrustZone, SELinux and a virtualization enabled implementation of the GlobalPlatform TEE API. The main contribution of this work is an analysis of the interrupt mechanisms of the ARMv8 architecture with TrustZone, along with a comparison of different possible implementations, which have been described, developed and benchmarked.

The benefits of the T-KVM are its flexibility (programmability of the guests and of the Secure World) and compatibility with the existing Cloud and smart devices architectures (GlobalPlatform and KVM/libvirt support). In fact, since T-KVM adapts and combines existing open source components which are already part of virtualized systems and OpenStack Cloud Computing infrastructure (i.e., KVM, libvirt, qemu, etc.), the support of T-KVM in these environments is straightforward to implement. Lastly, monitoring and (remote) attestation are eased by the combination of a standard hypervisor with TrustZone.

On the other hand, the lack of the ARM VE support in the Secure World does not allow the hardware assisted virtualization of the TEE. Nonetheless, it is possible to functionally implement multiple TEE (or vTPMs) using paravirtualization or virtualization at the application layer.

As for the analysis of the overhead introduced by the proposed solution, it is possible to claim that the performance cost of T-KVM is acceptable. In fact, even if the secure request overhead is significantly higher than a trap in the KVM hypervisor, the execution frequency of the former is expected to be lower than the latter: a service request is issued by a T-KVM guest only to control the communication between the guest and the Secure World, as all the data exchanges will be performed through shared memory. In regard to the FIQ latency tests, the results show that all software solutions to manage the FIQ exclusively handled at EL3 (i.e., Monitor only and Monitor/RTOS cases) introduce overhead which impacts on the FIQ latency. For this reason, in order to privilege the RTOS latency response time, the interrupt management implementation chosen by T-KVM is RTOS only.

Finally, the future work includes the implementation of a complete T-KVM ARMv8 prototype in the direction of automotive and avionics use cases such as infotainment or ADAS. With this in mind a full fledged version of the shared memory mechanism (Section III-A3), GlobalPlatform TEE (Section III-A2) and Trusted Boot (Section III-A1) will be developed, tested and benchmarked. Related to the interrupt management mechanisms, this work has given us the possibility to evaluate different approaches, of which the "RTOS only" has been selected as final solution. Unfortunately, it still lacks of a way to assign interrupts to the Monitor mode. This problem will be investigated in future works, exploring software methods to overcome this limitation of the TrustZone hardware implementation.

Of interest is also the support and integration of T-KVM in OpenStack, which would enable a complete new set of Cloud features based on real time and Trusted Computing services such as real time tasks allocated to the VMs, location aware scheduling of new instances, trusted multitenancy, etc.

ACKNOWLEDGMENT

This research work is an extension of the "T-KVM: A Trusted Architecture for KVM ARM v7 and v8 Virtual Machines" publication [1], awarded best paper at the IARIA Cloud Computing Conference 2015 and supported by the FP7 TRESSCA project (grant number 318036).

This research extension has been supported by the DREAMS project under the grant number 610640.

REFERENCES

[1] M. Paolino, A. Rigo, A. Spyridakis, J. Fanguede, P. Lalov, and D. Raho, "T-KVM: A Trusted Architecture for KVM ARM v7 and v8 Virtual Machines," IARIA Cloud Computing 2015 (CC2015), 2015, pp. 39–45.

[2] M. Wei, B. Heinz, and F. Stumpf, "A Cache Timing Attack on AES in Virtualization Environments," in Financial Cryptography and Data Security, ser. Lecture Notes in Computer Science, A. Keromytis, Ed. Springer Berlin Heidelberg, 2012, vol. 7397, pp. 314–328. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32946-3_23

[3] B. Saltaformaggio, D. Xu, and X. Zhang, "Busmonitor: A hypervisor-based solution for memory bus covert channels," 2013.

[4] G. Pék, A. Lanzi, A. Srivastava, D. Balzarotti, A. Francillon, and C. Neumann, "On the feasibility of software attacks on commodity virtual machine monitors via direct device assignment," in Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ser. ASIA CCS '14. New York, NY, USA: ACM, 2014, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2590296.2590299

[5] N. Elhage, "Virtunoid: A KVM Guest - Host privilege escalation exploit," 2011.

[6] S. E. Madnick and J. J. Donovan, "Application and Analysis of the Virtual Machine Approach to Information System Security and Isolation," in Proceedings of the Workshop on Virtual Computer Systems. New York, NY, USA: ACM, 1973, pp. 210–224. [Online]. Available: http://doi.acm.org/10.1145/800122.803961

[7] M. Paolino, "Isolating ARM platforms: towards secure virtualized embedded systems," February 2015. [Online]. Available: http://www.cspforum.eu/uploads/Csp2014Presentations/Track_8/Strong%20Isolation%20in%20ARM%20Platforms.pdf

[8] G. Chen, "KVM - Open Source Virtualization for the Enterprise and OpenStack Clouds," IDC, 2014.

[9] C. Dall and J. Nieh, "KVM/ARM: Experiences Building the Linux ARM Hypervisor," 2013.

[10] T. Alves and D. Felton, "TrustZone: Integrated hardware and software security," ARM white paper, vol. 3, no. 4, 2004, pp. 18–24.

[11] GlobalPlatform, "TEE Client API Specification, Public Release v1.0," 2010.

[12] Global-Platform, "TEE internal API Specification, Public Release v1.0," 2011.

[13] X. Xu, C. Xiao, C. Gao, and G. Tian, "A study on confidentiality and integrity protection of SELinux," in Networking and Information Technology (ICNIT), 2010 International Conference on, June 2010, pp. 269–273.

[14] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in multi-tenancy cloud," in Security Technology (ICCST), 2010 IEEE International Carnahan Conference on, Oct 2010, pp. 35–41.

[15] A. Vahidi and C. Jämthagen, "Secure RPC in Embedded Systems: Evaluation of Some GlobalPlatform Implementation Alternatives," in Proceedings of the Workshop on Embedded Systems Security, ser. WESS '13. New York, NY, USA: ACM, 2013, pp. 4:1–4:7. [Online]. Available: http://doi.acm.org/10.1145/2527317.2527321

[16] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The Untapped Potential of Trusted Execution Environments on Mobile Devices," Security Privacy, IEEE, vol. 12, no. 4, July 2014, pp. 29–37.

[17] C. Gehrmann, H. Douglas, and D. Nilsson, "Are there good reasons for protecting mobile phones with hypervisors?" in Consumer Communications and Networking Conference (CCNC), 2011 IEEE, Jan 2011, pp. 906–911.

[18] F. Liu, L. Ren, and H. Bai, "Secure-Turtles: Building a Secure Execution Environment for Guest VMs on Turtles System," Journal of Computers, vol. 9, no. 3, 2014, pp. 741–749.

[19] GlobalPlatform, "TEE System Architecture v1.0," 2011.

[20] M. Paolino, M. M. Hamayun, and D. Raho, "A Performance Analysis of ARM Virtual Machines Secured Using SELinux," in Cyber Security and Privacy, ser. Communications in Computer and Information Science, F. Cleary and M. Felici, Eds. Springer International Publishing, 2014, vol. 470, pp. 28–36. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-12574-9_3

[21] C. Macdonell, X. Ke, A. W. Gordon, and P. Lu, "Low-Latency, High-Bandwidth Use Cases for Nahanni/ivshmem," 2011.

[22] M. Paolino, "A Shared Memory zero-copy mechanism for ARM VMs:vosyshmem," February 2015. [Online]. Available: http://www.virtualopensystems.com/en/products/vosyshmem-zerocopy/

[23] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "seL4: Formal Verification of an OS Kernel," in Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 207–220. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629596

[24] R. Barry, "Real time application design using freertos in small embedded systems," 2003.

[25] "OPTEE," February 2015. [Online]. Available: https://github.com/OP-TEE

[26] "OSv," February 2015. [Online]. Available: http://osv.io

[27] J. Brunel, R. Pacalet, S. Ouaarab, and G. Duc, "SecBus, a Software/Hardware Architecture for Securing External Memories," in Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on, April 2014, pp. 277–282.

[28] M. D. Grammatikakis, "System-Level Modeling of a NoC Firewall on Spidergon STNoC," February 2015. [Online]. Available: http://www.cspforum.eu/uploads/Csp2014Presentations/Track_8/System-Level%20Modeling%20of%20a%20NoC%20Firewall%20on%20Spidergon%20STNoC.pdf

[29] "Programmer's Guide for ARMv8-A," March 2015. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.den0024a/DEN0024A_v8_architecture_PG.pdf

[30] "Programmable Interrupt Controllers: A New Architecture," July 2015. [Online]. Available: http://community.arm.com/groups/processors/blog/2015/07/27/gicv3-architecture

[31] "ARM Trusted Firmware," February 2015. [Online]. Available: https://github.com/ARM-software/arm-trusted-firmware

[32] "FreeRTOS v8.2.2 port (AARCH32) for ARMv8 platform (ARM FastModel virtual platform and ARM JUNO Development Platform) using the GCC ARM compiler (arm-none-eabi)," August 2015. [Online]. Available: http://interactive.freertos.org/entries/83649935-FreeRTOS-v8-2-2-port-AARCH32-for-ARMv8-platform-\\ARM-FastModel-virtual-platform-and-ARM-JUNO-Developm

[33] "Add a dispatcher for 32-bit Secure Payload," August 2015. [Online]. Available: https://github.com/ARM-software/arm-trusted-firmware/pull/363/commits

[34] E. Keller, J. Szefer, J. Rexford, and R. B. Lee, "NoHype: Virtualized Cloud Infrastructure Without the Virtualization," SIGARCH Comput. Archit. News, vol. 38, no. 3, Jun. 2010, pp. 350–361. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816010

[35] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the Hypervisor Attack Surface for a More Secure Cloud," in Proceedings of the 18th ACM Conference on Computer and Communications Security, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 401–412. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046754

[36] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes," SIGOPS Oper. Syst. Rev., vol. 41, no. 6, Oct. 2007, pp. 335–350. [Online]. Available: http://doi.acm.org/10.1145/1323293.1294294

[37] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky, "HyperSentry: Enabling Stealthy In-context Measurement of Hypervisor Integrity," in Proceedings of the 17th ACM Conference on Computer and Communications Security, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 38–49. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866313

[38] X. Ge, H. Vijayakumar, and T. Jaeger, "Sprobes: Enforcing Kernel Code Integrity on the TrustZone Architecture," CoRR, vol. abs/1410.7747, 2014. [Online]. Available: http://arxiv.org/abs/1410.7747

[39] R. Perez and et al., "vTPM: virtualizing the trusted platform module," in Proc. 15th Conf. on USENIX Security Symposium, 2006, pp. 305–320.

[40] R. e. a. Sailer, "sHype: Secure hypervisor approach to trusted virtualized systems," 2005.

[41] H. Nahari, "Trusted secure embedded Linux," in Proceedings of 2007 Linux Symposium, 2007, pp. 79–85.

[42] T. K. Lengyel, T. Kittel, J. Pfoh, and C. Eckert, "Multi-tiered Security Architecture for ARM via the Virtualization and Security Extensions," in Proceedings of the 2014 International Semiconductor Laser Conference, ser. ISLC '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 308–312. [Online]. Available: http://dx.doi.org/10.1109/DEXA.2014.68

The Dichotomy of Decision Sciences in Information Assurance, Privacy, and Security Applications in Law and Joint Ventures

Simon Reay Atkinson
Centre for International Security Studies,
FASS, University of Sydney,
Sydney, Australia
e-mail: simon.reayatkinson@sydney.edu.au

Gregory Tolhurst
Faculty of Law
University of Sydney
Sydney, Australia
e-mail: greg.tolhurst@sydney.edu.au

Liaquat Hossain
Information Management
Division of Information and Technology Studies,
The University of Hong Kong
Hong Kong
e-mail: lhossain@hku.hk

***Abstract*— Research and practice in decision sciences can be viewed from the dichotomy that exists in decision making and decision taking, where decision making is considered as consensus driven process and decision taking is considered as an act. We build upon a conceptual paper presented at INFOCOMP 2014 considering Law as different types of network and how an understanding of these networks, at the systems level, might assist in decision making and taking processes necessary for: information assurance; privacy; and security applications in Law – as may be applied in Cyber through emerging legal networks. We first identify the systems we might be working with before considering Law as a networked ecology. We then look at law beyond existing stable, more certain and ruled jurisdictions and how it might be applied to decision making and taking in Cyber. We consider an example of how law may apply in areas of uncertainty and where existing jurisdictional remits may no longer apply, e.g., in stateless jurisdictions or those impacted by instability and uncertainty following a disaster. We conclude by considering how Legal Networks may assist in the decision making, taking and social problem solving processes in Cyber and so contribute to system resilience.**

Keywords-Collaboration; Network Law; Stateless Jurisdictions; Fuzzy Logic; Ecologies; Good Faith.

I. INTRODUCTION

This paper considers Law as comprising different networks and how an understanding of these networks at the *systems* level might assist in the decision making and taking processes with particular application in addressing complex problem solving such as recovery from recession and in Cyber [1]. We first identify the systems we might be working with before considering Law as a networked ecology. We note that Europe has two different types of jurisdictional systems identified as *Common Law* and *Statutory / Codified* Law. We suggest that in recovering from recession, both these 'conceptual and normative tools [will be necessary] to [re]connect…Europe to its institutional design' [2]. Furthermore, having both *Common* and *Statutory* Law may provide a unique European *co-adaptive* [3] advantage by providing the essential *variety* [4] for complex problem solving. Regeneration of Europe without enabling interaction between the two codes would potentially 'exclude large groups of citizens from the political process, but also, in the long run, destabilize and delegitimize the European…project' [2]. As John Dunne [5] comments, 'if a clod be washed away by the sea, Europe is the less'. This paper looks at law as networks and the lacunae that exist between and beyond largely state-based jurisdictions, e.g., in Cyber. We consider how such an approach might be applied to better managing instabilities, such as containing or preventing an epidemic or recovery from recession. We identify examples of how law and civil infrastructures and their associated networks may interact. We conclude by considering *Jurisprudential Networks* and *Network Law* and how their ecology may exist with similarly entangled legal networks.

Combined, the authors are thematic leads in the areas of complex systems, contract law, digital and cyber ecologies, the management of knowledge including commercial law, restitution and dynamic social networks. The authors bring this knowledge to bear in the emerging area they posit to be 'Network Law' and 'Jurisprudential Networks'. Section II identifies the legal statutory and network systems and structures we may be working within before in the next section examining law as a network. We then consider Law where it presently stands and as it may be applied in areas beyond the state and thereby more certain jurisdictional controls and enforcement. Finally, we consider what may be termed 'Cyber-in-Law' and scope how such legal ecologies may emerge and may assist the decision making and taking process.

II. SYSTEMS IDENTIFICATION

Communications literature maintains that hierarchical structures provide a superficial representation of how work actually gets done [6]. Similarly, Stacey [7] posits that dynamic organizations should be viewed as a collection of

informal social networks (i.e., shadow structures beneath the formal structures); so allowing their elasticity to sustain continuous innovation and learning [8]. Also, taking Granovetter's [9] notion of the importance of strong and weak ties, we suggest the economic sociology of system identification and argue that weak signal detection could serve as proactive strategy for exploring 'Network Law' and 'Jurisprudential Networks' in cyber. Using this as a basis for system identification, we consider decision making and taking as to 'how work gets done in networks'; 'how work may be organizationally gradated within Law', and finally, in terms of the two predominant 'codes' of law.

A. Abbreviations and Acronyms

Within organizations and networks, we consider one of the underlying principles to be that of *trust* and the trusts established between networks to allow *systems* to work without being ordered to do so. These systems we contend extend to include Law and its application. As identified by Shaw [10]:

> Perhaps the most important general principle, underpinning many international legal rules is that of *good faith*. This principle is enshrined in the UN Charter, which provides in Article 2(2) that "all Members…shall fulfill in *good faith* the obligations assumed by them in accordance with the Charter".

Similarly, the International Court declared in the *Nuclear Tests* case [11], *inter alia*:

> One of the basic principles governing the creation and performance of legal obligations, whatever their source, is the principle of *good faith*. *Trust* and *confidence* are inherent in international co-operation [we call collaboration], in particular in an age when this co-operation in many fields is becoming increasingly essential. Just as the rule of *pacta sunt servanda* [agreements must be kept] in the law of treaties is based on good faith, so also is the binding character of an international obligation assumed by unilateral obligation [12].

These understanding of trust are very similar to those developed by Augustin José Menéndez where he states, *inter alia*:

> The first [*instrument*] is the instrumental inclusion of *trust*. From the political perspective, *trust* needs to be developed in the EU, to *legitimize* majoritarian and redistributive politics and strengthen center-periphery relations. *Trust* both enhances societal compliance with transnational norms of cooperation and conformity, and at the same time provides the *common* framework in which transnational cooperation enables the construction of social institutions. This is…the implicit trust and understanding that comes from a continent full of citizens that interact, on a continuous and *intuitive* basis. And that sense of *mutual trust* that comes from communication, and communication alone, can further stabilize both the European space and legitimize the Union's position in it [2].

Mumford [13] considered an important *risk* factor to be *trust*: 'because innovation is frequently a journey into the unknown, *trust* is a major factor in its successful assimilation'. Contrastingly, Giddens [14] defines trust as 'confidence in the reliability of a person, or system, regarding a set of outcomes or events' and Mumford further observes 'risk and trust are inextricably intertwined'. Considering *good faith* as combining *trust* and *confidence* and taking forward Mumford, Giddens and Mintzberg's [13]-[15] understanding, it is suggested that:

> '*Trust* may be a function of the Likelihood of a person or system being able to comprehend, explain, understand [risk] by logic and deal with a set of outcomes or events' [16].

Therefore, *Risk* may be considered as obverse to *Trust*:

> '*Risk* may be a function of both the Likelihood of an adverse event occurring and a system or person's ability to comprehend, explain and understand [risk] by logic' [16].

We posit (after Hossain & Wigand [12]) that organizations need to be seen as dynamic (elastic and plastic) *social-influence* networks (*SINners*!) In these collaborative [16] networks, complex operations (requiring tacit knowledge exchange [17]), are achieved through social (and in this respect, also cyber-) interactions beneath the formal hierarchical control structures. *Co-adaptive* [3] viability in maintaining operational effectiveness and efficiency [18] may therefore depend more on how we socialize and capitalize 'our' formal (hierarchical) and informal (social) networks to achieve shared common goals. In this paper, we consider *law* as a network applying both formal coordination by control and rule (CRC) and informal collaborative social influence (CSI) networks [19]. We further identify, building on work by Harmaakorpi et al. [20] a 'techno-socio-economic paradigm', aligning significantly to CRC networks, in which:

> '*Info/Techno-Socio* (ITS) systems seek to *program* (as opposed to *programme*) the relationship between technical processes and humans by digitizing performance *fidelity* and coding for repeatable *risk free* procedures in computer-control-spaces so that data and communication do not [temporally] contradict each other' [16].

Info/Techno-Systems [21] are seen to be ideal for achieving "in time" coordination by control and rule (CRC). By contrast Socio-Info/Techno systems are seen to be capable of enabling collaboration (CSI), "over time", in which:

> 'Socio-Info/Techno (SIT) systems stress the reciprocal interrelationship between humans and computers to foster improved *shared awareness* for *agilely* shaping the social programmes of work, in such a way that humanity and ICT [control] programs do not contradict each other' [18].

Based on this understanding of the Cyber combining both CRC / ITS and CSI / SIT networks, it is considered Cyber- may be defined as:

'A technologically bounded, largely immeasurable, strongly scientific, stochastic control space; comprising virtual-media and the display of data dealing with the *real* communication of *facts* and the *conceptualization* of other plausible possibilities, themselves capable of generating *strong* physical and *weaker* more social effects and *influencing* them' [22].

III. JURISDICTION AND JURISPRUDENCE

We consider *Jurisdiction* (from the Latin *ius*, *iuris* meaning 'law' and *dicere* meaning 'to speak') as the *practical authority* granted to a formally constituted legal body to make pronouncements on legal matters and to administer justice within a defined *legal environment*. It also refers to the inherent authority of a court to hear a case and to declare a judgment and the [sovereign] power to govern or legislate; make or enforce laws and the power / right to exercise authority in that *environment*.

We take a more specific understanding of *Jurisprudence* (*juris prudentia*) as being about the *ecology* of law, including its *cultural* and *social* underpinnings. In this understanding, we consider jurisprudence as acting in two interconnected ways:

1. *Interstitial* issues of law as a social organization and legal instrument relating to the local political, sûréte (considered in the French as including assurance, sureness, trusts, reassurance, safety and security) and economic (PŜE) [23] global social ecology in which it functions.

2. *Existential* issues of law as a *social institution* and *legal system* relating to the *global* political, sûréte and economic social *ecologies* in which it *functions*.

A. Statutory / Codified (Roman) Law and Common Law

We identify two predominant systems of law:

1. Common (Customary) Law is a *system* of laws originating from the English *Commonwealth* (or 'common weal / good') and based on court decisions, on the doctrines *implicit* in those decisions, and on *customs* and *usages* rather than on *codified* written laws. It is underpinned by a *jurisprudential* body of law responsible for *socializing* judicial *decisions* and *customs*, as distinct from those of *statute* law. Common-law courts base their decisions on prior judicial pronouncements rather than on legislative enactments. Under the doctrine of *stare decisis*, common-law judges are obliged to adhere to previously decided cases, or precedents, where the facts are substantially the same. Customary *practice* allows common law to *adapt* to the local *ecology*; at the same time, stare decisis provides certainty, uniformity, and predictability and makes for a stable jurisdictional *environment*;

2. Civil / Codified (Statutory) or Roman (Latin) Law is a legal system originating in Western Europe, intellectualized within the framework of 'late Roman law' (the Code of Justin overlaid by Germanic law and local *environmental* practices). The most prevalent feature is that its core principles are *codified* into a *referential* jurisdictional *system*, which serves as the primary source of law. This contrasts with 'common law systems' whose intellectual framework comes from judge-made *decisional* law giving *precedential* authority to prior court decisions. *Codified* or *Statutory* law is written (as opposed to oral or customary); set down by a legislature / legislator and approved by its law creating *jurisprudential* body. Conceptually, codified law proceeds from social *abstractions*; to formulate general *environmental* principles that distinguish *substantive* (formal / *statutory*) from *procedural* (informal / *customary*) rules. It holds case law to be secondary and subordinate to statutory law. Consequently, the judicial *ecology* is socially *inquisitorial* and *unbound* by precedent.

IV. LAW AS NETWORKS

From the above *systems* analysis it is possible to consider three different network *ecologies* operating across the law:

1. *Network Law* we consider to be: programmable / downloadable and to exist within current jurisdictions; connecting between existing jurisprudences and jurisdictions. It is codified / programmed entirely or largely by CRC / ITS systems, in which the main interaction is between IT, and IT and human users – with minimal involvement from the legal system, lawyers and solicitors.

2. *Jurisdictional Networks* we consider to 'have the authority and responsibility for making pronouncements on legal matters; administering justice within a defined *jurisdiction*; declaring judgments; legislating and enforcing laws *in time* within that *environment*. They are a distinct *entity* or *being* contained within existing jurisdictions and *connecting* between them and different jurisprudences – and which may create and have value by combining / *synthesizing* the existing historical legal codes, for example Common and Customary Law'.

3. *Jurisprudential Networks* we consider to be: '*entities* and *beings* with a responsibility for understanding the social and cultural underpinnings of the law. Over time these networks influence law and allow it to adapt to change, they promote *collaboration*. The concern of such networks is with law as a *social* organization and law as a *social* institution'.

A. Jurisdictional Networks

We consider legal networks as they may be applied through Common and Statutory legal systems through the associated executive, legislative, judicial and enforcement bodies. In this respect, we identify four *hard* coordination, rule and control *jurisdictional* networks: the *executive*; the *legislative*; the *judicial* and *enforcement*. In democracies, the executive is provided by the elected ruling party and the legislative by parliaments elected to hold the ruling party to account and to legislate. This forms the *legislative*

jurisprudence. Responsible for implementing (the statutory legal system) and interpreting (the customary legal system) laws and connecting between the executive, the legislative and enforcement bodies is the judiciary. This forms the *judicial* jurisprudence. The third jurisprudence is provided by those responsible for enforcing civil legislation – which in most states includes policing, taxation, border, health, defense and social services administration. This is suggested to be the *enforcement* jurisprudence. Figure 1 situates the different legal 'beings' as vertically integrated, with the *public* jurisprudence – the conversation of public opinion and consent – lowermost. Also shown are the two different codes of law: one, Codified / Statutory Law, which is more top down; the other, Common / Customary Law, which is more rhyzomic. Significantly, the *judicial* jurisprudence in both codes interprets and makes *social sense* of the law either through *inquisition* (Codified) or *precedence* (Common).

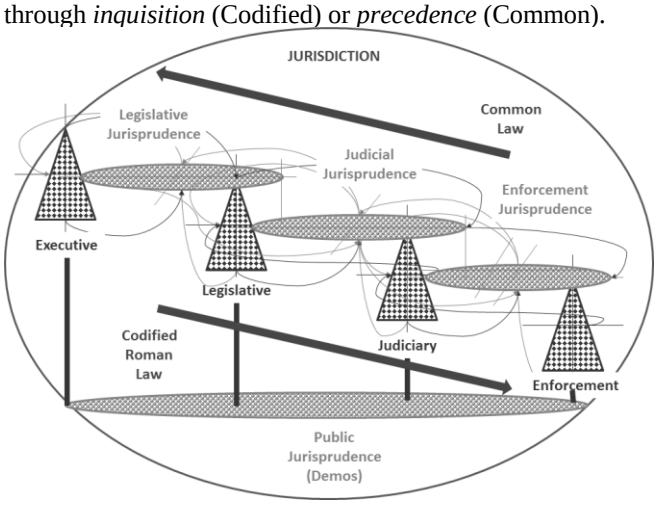


Figure 1. Jurisdictional and Jurisprudential Bodies

B. Jurisprudential Networks

We can identify three principal jurisprudential networks, the legislative, the judicial and enforcement, see Figure 2. At first glance this appears similar to the jurisdictional networks we identified. We do recognize that their responsibilities overlap. However, the jurisdictional networks are concerned with coordination and control (rank), while the jurisprudential networks are concerned with collaboration and influence (position). Examined from a horizontal perspective, *jurisprudential* responsibilities may be considered more in terms of position (than rank) and overlapping areas of responsibility. Significantly, this view also situates the Law within its *civil*, public and *social* settings. The *inquisitorial* and *precedential* interpretative roles of judicial jurisprudence also become clearer. Judicial jurisprudence connects between both *legislative* and *enforcement* jurisprudences. Specialist *soft* (informal) jurisprudence networks are identified to exist between the legislative and the judicial and the judicial and enforcement networks. We call these *Statutory* and *Customary* Jurisprudences. From a *Customary* and *Statutory* Law position, this analysis also identifies the priority given to the different judicial environments. Under *Statutory* Law, precedent is given to formal / codified rules and then to informal / customary ones. The position is reversed under *Common* Law, which gives precedent to informal *customs* and then to formally *codified* laws (the principle of *stare decisis*).

This research reinforced the position that 'for understanding and implementing cross-jurisdictional decision *making* and *taking* one needs to understand the different jurisprudences'. More precisely, one needs to interact at the jurisprudential level between both codes and specifically with the *statutory* and *customary* jurisprudences. This is not always well understood – for example, the continuing struggle between the English Courts and British Parliament in implementing European Court of Human Rights statutes. Most significantly, it is the social and collaborative *jurisprudential* networks that enable the Law to be *seen as*, *shared* and practiced *justly*.

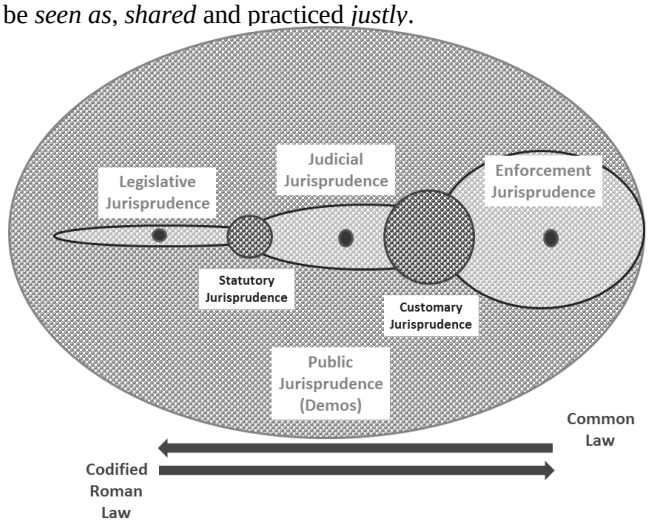


Figure 2. Jurisprudential Networks

V. Decision Making and Taking

At its heart, decision-taking is about the decision-making process – how, who, what, where and when. In this ideal world, strategy is primarily about 'observation' and 'orientation', while 'decision' and 'action' are best left to tacticians and operatives. However, in the 'real' world, strategists have to take account of all the factors impinging upon their strategic environment and no strategist can possibly operate in isolation – there is a social and network component to their knowledge, underpinned by the (social) strategic planning processes and the (personal) cognitive ones [24].

As connectivity and the availability of information has increased, this has often impacted negatively upon the ability to take and to make effective decisions. The science of decision making and taking examines the basis of effective decision-making and decision-taking in complex systems so as to adequately differentiates between the two [24].

There is a morality / ethicality to the decision making and taking process that is not always understood and rarely

articulated [24]. Considering Boyd's simple OODA Loop (Observe, Orient, Decide, Act) [25] there are essentially two loops contained within the one. One loop (Loop 1) is the observe-orient-decision-make loop; the other the decision-take-act loop (Loop 2). Together, arguably, they preserve a moral and ethical basis with decisions being made and taken based upon the available facts and the three relatives (3Rs: time, timing and tempo):

Loop 1 may be the home of the diplomat, the public servant, the researcher, designer and planner [26]. Loop 1 can be described in terms of its focus upon the methodology, on managing the loop from observation (experimentation, for example) through to orienting the structure appropriately for a decision to be made. The danger in Loop 1 is its focus on the levers and structures of power not necessarily the agency / and agents necessary to implement and carry out its decisions or inform its designs [24].

Loop 2, by contrast, concentrates on decision-taking and action with no previous research or observation, scant regard for theory and philosophy and believes largely in the delivery of action through agency / agents in order to exploit the results. This is the home of the Neo-Cons (advocates of the use of force, manipulation and deception of national / international affairs to promote rapid (democratic) political change / adoption of free-market policies; including by military means), who focus on action as a means of changing the status quo in their favor and breaking existing structures, methods and processes they see as constraints to their behavior. Their emphasis is on controlling the perception and the narrative as a means of coordinating and dictating the process and methodology [24].

In an adaptive ecology, one would expect the decision making and taking process to be continuous. After Bunge [27] (who considers knowledge as social), the collaborative social, *decision making* phase may be described more by CSI / SIT networks, while the *decision taking* phase may be described more by coordination, rule and control (CRC / ITS) networks. In a legal setting, it may be suggested that the *jurisprudential* networks provide for reflection and adaptation and the *jurisdictional* networks the necessary order for coordination and control. This recognizes work by Gray [28] and Luttwak [29] 'that places emphasis on the importance of *strategic culture* in *networked social processes* and which underpin planning, *decision-making* and so *decision-taking*: good decisions are not capability driven' [30]. It is often these reflective, social networks that are *sacrificed* to *optimization* regimes that concentrate on *objective* metrication [18].

VI. Cyber-in-Law

Zadeh [31] noted *decision making* and *taking* has been dominated by Probability Theory, while Clark et al. [32] suggested that 'a new mathematical model, based upon vagueness, fuzzy sets and partial possibilities [dealing with uncertainty], may be required to advance the science'. Additionally, Pólya recognized the relative ease of statistical programming for verification 'has tended to favor the heuristic [evidence based] reasoning of the mathematician rather than the inductive reasoning of the physicist' [33].

Cyber may be seen to consist of both the internet and the social networks that the internet supports; connecting between two poles. One sub-system may be identified and classified as being by **"**Coordination Rule and Control (CRC)" (akin to Network Law) (*explicit*); the other described as being through "Collaboration and Social Influence (CSI)" (akin to Jurisprudential Networks) (*implicit*) [34], [35]. These system attributes provide the necessary and "requisite variety" [4] to enable both control, "in time", e.g., Just In Time (JIT), and influence [36]-[40], "over time".

Our research indicates that understanding the connections between these poles involves *Fuzzy Logic* (FL). Emerging from Probability Theory (PrTh) with its binary logic-sets Zadeh [41] put forward Fuzzy Logic where 'linguistic variables with a truth value ranging in degree between 0 and 1 may be 'managed by specific functions'. Its main conceptual difference with PrTh, is that *Fuzzy Logic* considers degrees of truth; *vagueness* (in terms of lack of specificity and not knowing precisely); *partial truth*; *partial possibility* [42] and *uncertainty*. Whereas, standard Probability Theory deals with the stochastic – thereby global – partitioning of certainties; not the understanding of partial possibilities or partial truths:

'Viewed through the prism of partiality, probability theory is, in essence, a theory of partial *certainty* and random behavior. What it does not address – at least not explicitly – is partial truth, partial precision and partial possibility – facets, which are distinct from partial certainty and fall within the province of fuzzy logic. This observation explains why PrTh and FL are, for the most part, complementary rather than in competition' [31].

Noting the linkage between PrTh and FL since the 1990s Zadeh [31], recognized: 'the concerted drive toward automation [and control] of decision-making in a wide variety of fields [e.g., Cyber]…A side effect…is the widening realization that most real-world probabilities are far from being *precisely* known or *measurable* numbers'. Tong [43] had previously concluded that: 'Fuzzy models can be made to work…and, even in more complex situations (more variables or less data for example) they could capture basic behavior'. He considered them relatively simple to construct, being themselves quite simple structures whose greatest value lay in communicating process to others, where the linguistic value of a highly complex [Bayesian] model is doubtful. Tong went onto to suggest that fuzzy models are perhaps 'most valuable as tools for understanding basic characteristics rather than as detailed descriptions of process [and control] behavior'.

In law we may consider a road speed limit as an example of compliance / control by reason of certain sanction. Generally, people obey for fear of a fine if caught going over the limit [44], and the speed limit may result in a reduced number of accidents caused by speeding. We do not question the need for formal hard rules; every network needs such rules to operate *efficiently* [45]. A concern may be the extent to which it is possible to promote good behavior, including

in Cyber and beyond state-based jurisdictions, based simply on Law. The set speed limit may not promote responsible driving; it may simply ensure people do not go over the speed limit; indeed, it may simply promote driving at the speed limit in all situations, regardless. Traffic conditions vary for many different reasons requiring drivers to *make* and *take* decisions about speed. In this case we are dealing with a complex system, for which a hard rule cannot regulate behavior. Hence, as noted, the resort to more fuzzy concepts [46] for dealing with *uncertainty* in more complex ecologies, such as exists in Cyber. We posit that it is the *trust* and confidence of CSI principles that are central to *influencing* people to act in a good and collaborative way – particularly in areas of uncertainty where reflective *learning* plays a key role. In saying this, we do not doubt that well-formed principles of CRC / ITS may help, particularly as regards to enforcement and providing guidance as to *fail-safe* protocols and procedures. We also note, though, that even enforcement agencies are influenced by CSI / ITS principles as they, too, are parts of the jurisprudential networks.

VII. Stateless Jurisdictions and Changing Ecologies

Strong signal controls drown out the *weaker signals* necessary for innovation and adaptation in a number of ways. The most significant way they do this is to create norms-of-behaviour that tend to award compliance to rules and conformity and punish those who think differently, are awkward and challenge the consensus [24]. This is where *managed diversity* (for all its obvious goods) trumps the requirement for *complex variety* [35], necessary to innovatively problem solve and control [4], [47], [48].

A *weak signal* decision mathematically can have the same strength as a *strong signal* decision; however, it also has some very different and unique characteristics [24]. Because of the lack of *resistance*, questions of *time*, *timing* and *tempo* vary – in other words, there is limited resistance to be overcome and, indeed, the main challenge is to allow for *reasoning* and the *reflective capacity* [49] – necessary for *episteme* (*making* possible the structures / apparatus necessary for *taking* a decision) based, not on the separation of the true from the false, but upon what may, or may not, be characterized as *empirical* [50].

Because of the lack of *knowing*, it is not so much about providing persuasion as providing the reasoning for making a decision, informed very often more by *intuition* (the ability to comprehend without inference and / or the use of reason). Because of the weak signals and potentially *fleeting* nature of the decision to be *taken*, it is not a question (as for strong signal decisions) of doing one activity in isolation / or by constraining the other, e.g., freezing persuasion and reducing resistance. Decisions in the *weak signal* context need to have in place the *structure*, agents and agency necessary for *reasoning* – and for both *episteme* and *intuition* [24].

In two recent areas of research, the authors applied principles of *uncertainty*, *instability*, and *good faith* in relation to *statelessness* in terms of a Joint Venture case analysis and a green star accreditation system for building processes. In this respect, we see *statelessness* relating to the law; to nation states; the cyber and social / physical states of matter – or Metaphysics. In conditions of statelessness and of high uncertainty and instability, there exists the possibility of phase changes as new emergent structures condense to articulate and define new beings.

Uncertainty applies to probabilities, as in a Risk Register and to physical measurements that are already made, or to Donald Rumsfelds' *known-unknowns*, *unknown-knowns* and *unknown-unknowns* (US DOD news briefing, 12 Feb. 2002). Specifically, we consider *Uncertainty* to:

> 'Arise in partially observable, opaque, stochastic environments / non-ergodic (complex) ecologies, overly prescribed, ruled or controlled (ergodic) regimes as well as due to lack of assurance, *instability*, ignorance and / or lack of caring and shared awareness; including indolence' [51], [52].

Instability can create Uncertainty and Uncertainty can create Instability but they are not the same thing, Instability may be considered as:

> 'The quality or state of being *unstable* and / or the tendency to behave in an unpredictable, changeable, *uncertain*, or erratic manner' [51], [52].

We suggest that the *entangled* nature of *trust*, *risk*, *uncertainty* and *instability* become apparent through the lenses scoped above. With respect to a joint venture (JV) or collaborative process, we may consider it as a journey into the unknown, where exploration and innovation is required that may not be prescribed, ruled or controlled. At its heart, therefore, a JV collaboration may necessarily be based upon good faith which we posit (in this regard) as being:

> An expression of implied intent to explore the establishment of a *trusting*, *collaborative*, shared aware relationship between one or more parties involving the temporal suspension of *disbelief* that one party may have towards another party's raison d'être, rationale, modus operandi, concepts or ideas. It especially involves having all parties thinking that the other party's intentions are *certain* (within the kirk established), benevolent, *trust-worthy*, competent, good, honest or true.

We reconstructed a famous case in New South Wales, Australia involving a Joint Venture, *Coal Cliff Collieries Pty Ltd v Sijehama Pty Ltd* [53]. The case involved a number of negotiations commencing in the mid-1980s and culminating in an ultimately unsuccessful JV some 5 years later. Our reconstruction of events was based upon factors and relationships considered to influence a given joint venture (represented by a matrix, S), the permanent function of the matrix is an indicator of the level of uncertainty associated with a specific stage the process [54], [55]; based upon similar work by several researchers [56] used in calculating the uncertainty of a Performance Measurement (PM).

With relation to this *Coal Cliff Collieries*, a 5×5 matrix, S, was developed using a Likert type [57] scaling to represent the major parties and their partners in the joint venture along with the existential factors that we presume created the conditions for seeking such an agreement in the

first instance. From this a Decision Measurement (DM) was assessed based upon levels of uncertainty at each stage of negotiation for the Heads of Agreement (HoA) and the Joint Venture, the HoA was expected to deliver, see Figure 3.

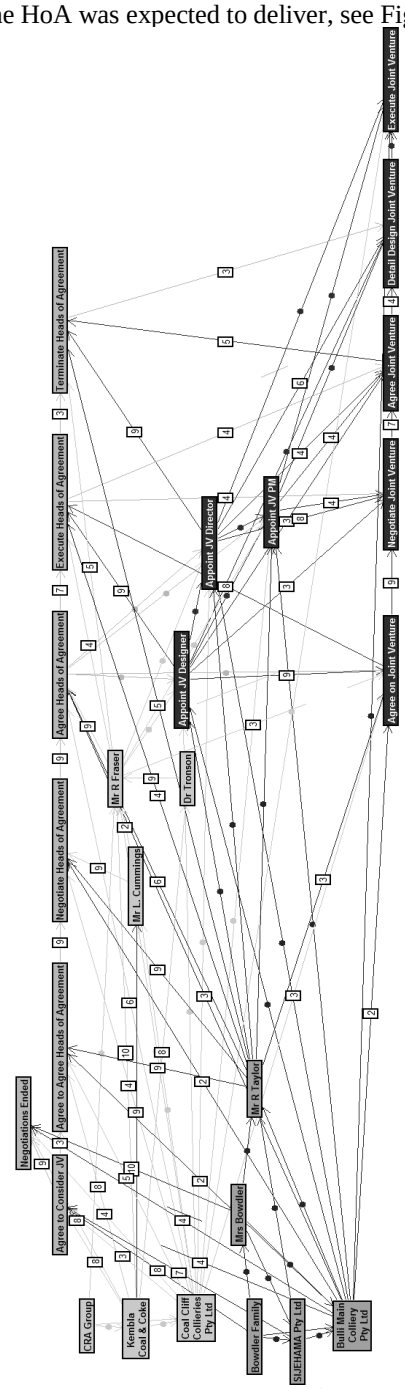


Figure 3. Longitudinal Concurrent Heads of Agreement and Joint Venture processes with downstream appointment of Joint Venture Designer; Joint Director and Project Manager (read bottom to top)

We determined that the JV negotiations broke down not through lack of rules and processes but due to the lack of trusts necessary to enable collaboration and shared awareness and so collaboration through an *uncertain* decision making / taking exploratory process.

By inserting trusted agents at key moments of the negotiation to better represent all parties (rather than applying rules and processes) it was found that uncertainty in the process could be 'brought under control' and, as a result, a satisfactory conclusion arrived at. In this instance, rather than collapsing to high degrees of uncertainty, the JV may have ended successfully in a new company / enterprise entity forming, see Figure 4.

In a separate study [58], see Figure 5, we considered two parallel (essentially disconnected) processes in the building industry, the build process; and, the Green Building Council for Australia (GBCA) Green Star programme for 'developing a sustainable property industry for Australia by encouraging the adoption of green building practices'. The aim (or mission) of the GBCA is to: 'develop a sustainable property industry for Australia and drive the adoption of green building practices through market-based solutions'[1].

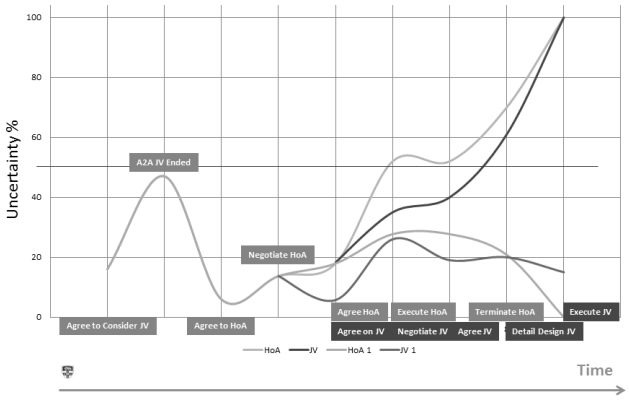


Figure 4. Heads of Agreement and Joint Venture Concurrent Processes Uncertainty Index, read left to right

Research was supported through the interview of over 25 building industry subject matter experts, considered levels of uncertainty at each Decision Measurement (DM) stage.

From Figure 6, it will be seen that, whereas the build process commences with a relatively high DM Uncertainty Index and that this increases initially, uncertainty in the process then reduces to a more manageable banding around the 15-20% uncertainty level.

The GBCA process, by contrast to build and design, appears much more *unstable*, with wide *uncertainty* swings of between 10-75% and with limited stability from one stage to the next. For planning purposes, this suggests that the current system may be unmanageable – without a foundation to predict (due to its inherent uncertainty) and / or to influence (since there is limited opportunity for collaboration) outcomes. For example, the GBCA Design Award is assessed at 50% *Uncertainty* – a basis of prediction that would be as precise for tossing a coin as following due process. Unlike the Build process, which creates opportunities for local, collaborative delegation of responsibility and authority, the GBCA process holds control

[1] Green Building Council for Australia (GBCA) Website, http://www.gbca.org.au/ visited Aug 2014.

through a limited number of players that largely disables feedback, while requiring reach-back to individuals and records (not held by the process) as the project develops.

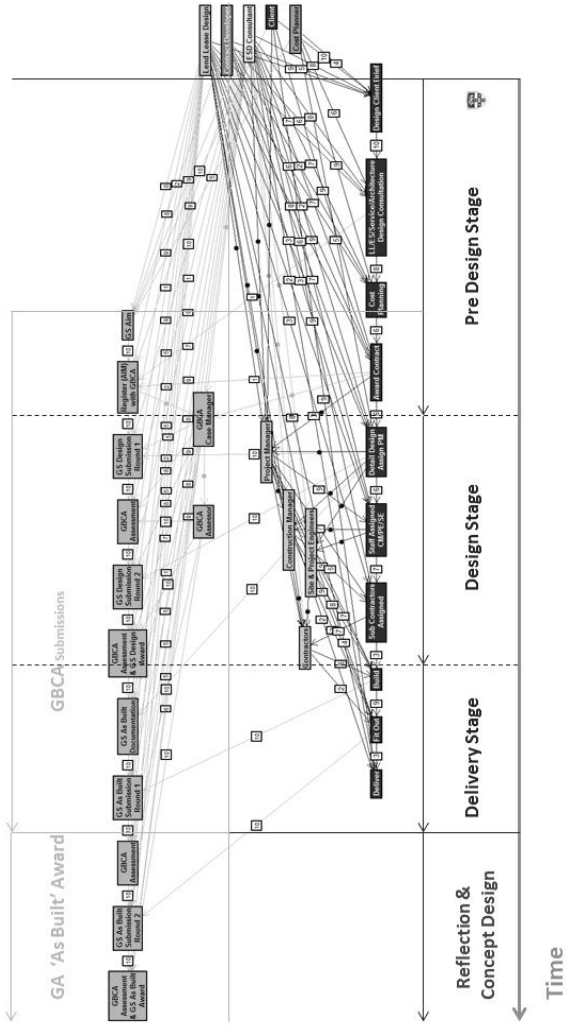


Figure 5. Build and GBCA System Processes (read top to bottom)

In addressing the failures of government and collective (collegial) intelligence prior to 9/11 and the Iraq War, the US 9/11 Commission [59] and the (Lord) Butler Enquiry [60] identified that overly controlled or formalized organizational structures such as those existing before 9/11 had not simply atrophied but had become 'tuned out' – no longer able to select between the vital weak-signals of innovation, adaptation [61] and change [62] (as threat or opportunity) [9] and the strong-signals of method [63] and process [64]. Recommendations arising from 9/11 [59] and the Global Financial Crisis were three fold: first has been to require greater transparency, for example, between the banks, investors, borrowers and governments; secondly, has been to demand greater regulation and thirdly, to move away from the need to know control model towards what has been described as the three needs model – need to know; need-to-share; need-to-use (3NM) [17].

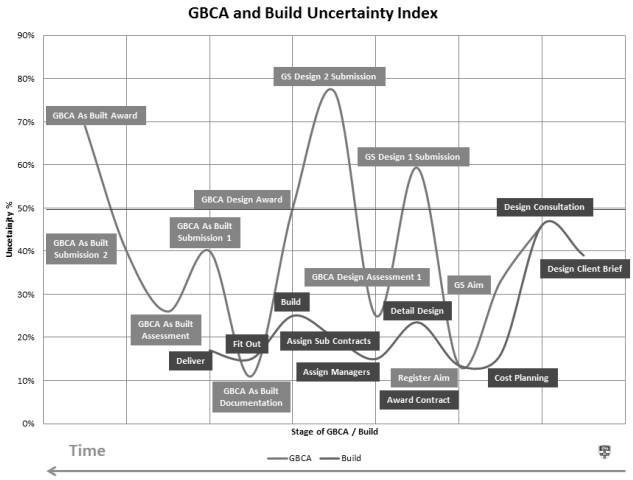


Figure 6. Build and GBCA System Uncertainties, read right to left

A control system identifies noise as risk and seeks then to remove it. Similarly, administrative processes and legal constructs based upon certainties seek to remove the noise – or uncertainty – from the system to achieve a degree of certainty against which legal judgments – based upon facts after the event – have been established. In a social setting, however, these same uncertainties can represent both noise but also the weak signals of innovation and change – potentially, as in 911, of threat also. *Uncertainty* – or rather manageable levels of uncertainty – can consequently be a good; enabling change and adaptation over time by testing the organization. If that ability to adapt or change by testing the ecology – in this case the ecology of a coal mine – are removed or constrained, then the ability for innovation, exploration, change and adaptation may also be impaired.

We readily accept that there are many examples of where people obey hard legal rules with the clear intention to comply with the law either because it is the law or because of the sanctions that may be imposed on them for contravening the law. Indeed such compliance may produce good results. A road speed limit is a typical example of such compliance by reason of sanction, people generally obey the limit for fear of a fine if caught going over the limit [44] and the speed limit may result in a reduced number of accidents caused by speeding than would be the case if people were left to judge for themselves the speed they should drive at in any particular circumstance. Moreover, we do not question the need for formal hard rules; every network needs such rules to operate efficiently [45]. Our concern is with the extent to which it is possible to promote good behavior through law. The set speed limit may not promote responsible driving; it may simply ensure people do not go over the speed limit, indeed it may simply promote driving at the speed limit in all situations. Traffic conditions vary for many different reasons requiring drivers to make decisions about speed, a hard rule cannot regulate behavior here as it is a complex environment and hence, as noted, the need to resort to more fuzzy concepts to promote thoughtful and good behavior whether it be driving or regulating a joint venture [46].

Collaborative processes, such as JVs, are a journey into the unknown and therefore require a degree of *trust*, *stability* and *certainty* in the proceedings. At the same time, an overly-prescriptive legal context may well deny those trusts forming in the first place and so the stabilities necessary to 'let go' and move to the next stage. As a result, the joint venture becomes fixed and frozen and can no longer move up or down – like a sailor afraid of heights clinging to the mast as their ship closes the rocks. We suggest that inherent within leadership and management is the ability to manage uncertainty and to identify those points of stability and agreement against which progress can be safely navigated [45].

Uncertainty above 50% would appear to be unmanageable and so, if not already, an unstable basis upon which to negotiate [54], [58]. Particularly a negotiation that, at some point, needs to focus on the identification and classification of its ecology; its standardization and ultimately upon the optimum and efficient delivery of a new company or organization – in this case a colliery. Optimizing upon uncertainty would simply be nonsensical and highly unstable – like our sailor letting go of three limbs and holding on with the fourth! At the same time, absolutely no uncertainty and no noise or dynamics within the system would appear to rule out any opportunity for innovation, from which change and adaptation might occur. This would suggest, empirically, that successful joint venture / collaborative processes may be those that seek to allow for uncertainty between, say, 15 and 40% and establish the structures necessary to allow for trusted debate in such a context [54], [58].

In management settings, we can identify two forms of leadership and management structures: the one more vertical and aligned with control type hierarchies typical of industrial / clockwork armies, such as fought in 1914; the other more horizontal, based upon trust and collaboration [24], [45]. In joint venture / award processes (such as GBCA), there is a need for a higher degree of collaboration and trusts to deal with the uncertainty and so create points of stability against which a successful venture may be founded. In other management structures, this might count as noise and so disrupt highly optimized manufacturing processes. The problem appears not to be controls or the law, per se, but who, where and when we choose to apply it.

The problems to be solved are very often not static and do not yield to a technique of 'thinking about it for a long period of time' because the ground upon which it sits continuously shifts. The application of complex theory provides valuable insights to how decisions are made and taken and importantly how to influence decisions. This is of vital importance in areas of human endeavor that defy regulation solely by command and control type regimes / processes. In particular, we are thinking of spaces that might be termed 'Stateless Jurisdictions' and which we see as operating according to principles of 'Network Law'. In these conditions of statelessness and of high uncertainty and instability, there exists the possibility of phase changes as new emergent structures condense to articulate and define new beings. Beings such as a successful joint venture that form from stateless or near stateless combinations to create a new entity. We maintain that the mathematics and legal conditions / rule-based processes enabling the emergence of such entities is entirely antithetical to those pertaining for 'steady state', stable type regimes. In fact, as argued in this paper the rigid application of 'steady-state' conditions and constraints can destroy the bases for such condensations to form. It is in these stateless spaces that we need to be able to encourage good behavior and trusts to form in a way that is to the benefit of the whole rather than simply forever punishing bad behavior and, ultimately, encouraging further the 'flight to cyber'.

In studies (one *dynamic* (*GBCA*) and one *retrospective* (*Coal Cliffs*)), it was possible to identify when and where uncertainty and thereby instability might occur [54]. In the event of the Coal Cliffs JV, it was shown that by introducing collaborative agents at a key point in the negotiations, that this could be instrumental to successful delivery of the JV and completion of the Heads of Agreement. In this respect, it may be possible to instrument negotiations in such a way as to forecast or even predict when interventions may be necessary to reduce uncertainty and so improve instability. In the GBCA study [58], through dynamic social network (DsN) and uncertainty analysis, it was possible to examine both the build and GBCA Networks and to make recommendations to improve collaboration / shared awareness. This would involve sharing trusts and thereby risks between both parties and, at the same time, creating a knowledge hub or Librarian type position that would retain knowledge of the process and previous builds, over time – rather than re-inventing the wheel, every build. This was contrary to industry expectations, which had posited improvements might be required within their communities, alone [58]. Research also pointed to an interesting dynamic relationship between *uncertainty* and *efficiency*. Industry was looking for efficiency gains and not understanding why the process could not be made more efficient. Research indicated that uncertainty in the process (see Figure 6) simply made any attempt at optimization untenable – it would be a bit like optimizing on moving ground. The process could only be made more *stable* and less *uncertain* by improving collaboration between the parties [58]. Only then, the system could be *optimized* and so made more *efficient*. Provided, of course, *optimization* did not remove the agents responsible for the retention and management of knowledge (which is not the same as KM!), so typical of Performance Management regimes.

The science dictates that hard controls and rules will always fail as emergent networks are too fluid and their rapid ability to adapt and change will simply by-pass static rules and regulations [54]. Regulation of behavior in such jurisdictions or the influencing of behavior that promotes the making and taking of good decisions for the individual and the network requires the harnessing of legal / process standards that promote reflection and an understanding of how they can be deployed throughout a network. This is the subject of future work.

VIII. NEW ECOLOGIES

We consider that in an *adaptive* system, the decision making and taking processes are continuous and part of an ecology constantly *testing* for both success and failure – so as to avoid catastrophic degradation (the deleterious / undesirable deterioration of a ecology; including destruction of ecosystems and extinction of system-networks). The law and certain process regimes (often used / applied by Government on industry / the private sector) can be seen as a fixed immovable, post-hoc (after the event), *metricable* [measurable] object, like a castle. Examined from a *jurisdictional* point of view, the *objective* of law / processes such as the GBCA [54], [58], can be seen as 'controlling in order to rule' based upon the *representation* of evidence (data). The means have become the ends and the jurisdiction drives the strategy. What constitutes jurisdictional or process *knowledge* in law and control-engineering is not the same as what constitutes *knowledge* in strategy and so decision making and taking [24]. Strategic knowledge in Law is vested within its *jurisprudential* social networks as it is within the social *techné* (expert 'know how'; subjective knowledge of how to 'changes things') and *phronesis* (reflective wisdom, which provides plausible explanation and guidance in times of uncertainty') contained within any successful organization [54]. It is this co-*adaptive* knowledge that is so important in understanding decision making and taking [24].

We contend that there is a need in the 21st Century, to 'put humanity back in the loop', and that people will be employed more often in those *complex* lacunae where no amount of control, rule or coordination will *make* sense. We also see these as being the vital decision making and taking *commons* fundamental to delivering timely laws; design; strategies; and, policies that will prevail / pervade 'over time'. We also recognize that *resilience* does not come from the info-techno-socio control type networks but from investment in *socializing* and capitalizing our socio-info-techno influence networks. One cannot understand these *complex* systems without understanding their underpinning networks and how they are managed and *controlled*; *influenced* and *led* [24]. Understanding how Law interacts at the project, unit, *jurisdictional* and systems *influence* and *jurisprudential* levels is therefore important. Not simply to aid understanding in times of crises, but to provide sustainable future programmes and to enable timely, collaborative, social responses to shocks and uncertainties, be they human-made or natural.

We consider Network Law as a *hard* entity contained within existing *Jurisdictional Networks* and *connecting* through IT between them and different jurisprudences [54]. We suggest that they may have specific value in combining and synthesizing historical legal codes, such as Common and Codified Law. We do not advocate new laws, for example, for Cyber, but for improved understanding and the establishment of connecting *soft* networks – hence, *Jurisprudential Networks* – to better socialize connections between existing jurisdictions, *Network Laws* and the cyber-internet – specifically in areas of uncertainty and potential instability, for example, a collaborative process of Joint Venture [54], [58]. It is in the area of Cyber and Law that this paper makes a contribution and which, based upon the principles derived and outlined in this paper including for Fuzzy Logic and Fuzzy Law, which the authors are taking forward for application and future development.

ACKNOWLEDGMENT

We acknowledge Lend Lease and the Royal Australian Navy and in particular the Faculties of Law, Arts and Social Science, the Centre for International Security Studies and Engineering and Information Management & Technology at the Universities of Sydney and Hong Kong.

REFERENCES

[1] S. Reay Atkinson, G. Tolhurst, and L. Hossain, "Decision making and taking in changing ecologies considering network law," The Fourth International Conference on Advanced Communications and Computation, INFOCOMP, July 20 – 24, 2014, Paris, France: IARIA, pp. 76-82.

[2] A.J. Menéndez, "Review of Developments in German, European and International Jurisprudence," Special Issue – Regeneration Europe M. Hartmann, and F. de Witte, Eds., German Law Journal: Berlin, pp. 441-712, 2013.

[3] A.-M. Grisogono, "Co-adaptation," Proceeedings of SPIE – the International Society for Optics and Photonics, 16 January, 2006, vol. 6039, article no. 603903.

[4] R. Ashby, An Introduction to Cybernetics. London: Chapman and Hall,1957.

[5] J. Dunne, "Meditation XVII: nunc lento sonitu dicunt, morieris," John Dunne (1572-1631), English, Roman Catholic convert to Protestantism, Lawyer, Diplomat, Poet, Vicar and Prolocutor to King Charles the First (of England). England: unknown, 1632.

[6] E. Stacey, "Collaborative Learning in an Online Environment," Journal of Distance Education. 14(2), pp. 14-33, 1999.

[7] R.D. Stacey, Complexity and Creativity in Organizations. San Francisco, CA: Berrett-Koehler Publishers, 1996.

[8] L. Hossain and R.T. Wigand, "Understanding virtual collaboration through structuration," in Proceedings of the 4th European Conference on Knowledge Management, 2003, pp. 475-484.

[9] M. Granovetter, "The Strength of Weak Ties," American Journal of Sociology, 1973, vol. 78, iss 6, pp. 1360-1380.

[10] M.N. Shaw, International Law. 4th Edition. Cambridge, England: CUP, 1997.

[11] I.C.J., "International Court of Justice (I.C.J.) reports," ICJ Reports quoting International Law Research (ILR), pp. 253, 257-267, ILR p. 398 and p. 412. 1974.

[12] L. Hossain and R.T. Wigand, "ICT Enabled Virtual Collaboration through Trust," Journal of Computer Mediated Communication, JCMC 10 (1) November, pp.22-31, 2004.

[13] E. Mumford, "Risky Ideas in the Risk Society," Journal of Information Technology, vol. 11, pp. 321-31. 1996.

[14] A. Giddens, The Consequences of Modernity. Cambridge, England: Polity Press, 1990.

[15] H. Mintzberg, D. Dougherty, J. Jorgensen, and F. Westley, "Some surprising things about Collaboration - knowing how people connect makes it work better," Organizational Dynamics. Spring, pp. 60-71, 1996.

[16] S. Reay Atkinson, A.M., Maier, N.H.M., Caldwell, and P.J. Clarkson, "Collaborative trust networks in engineering design adaptation," International Conference of Engineering Design, ICED11, 2011. Technical University of Denmark, Lyngby.

[17] S. Reay Atkinson, S. Lesher, and D. Shoupe, "Information capture and knowledge exchange: The Gathering Testing and

assessment of Information and Knowledge through Exploration and Exploitation," 14th ICCRTS: C2 and Agility, CCRP, 2009, Washington.

[18] S. Reay Atkinson, A. Goodger, N.H.M Caldwell, and L. Hossain, "How Lean the Machine: how Agile the Mind," The Learning Organization. vol. 19, iss. 3, pp. 183 – 206, 2012.

[19] D. Walker, S. Reay Atkinson, and L. Hossian, "Counterinsurgency through civil infrastructure networks," Second International Conference on Social Eco-Informatics (SOTICS) October 21 – 26, 2012, SOTICS: Venice.

[20] V. Harmaakorpi, I. Kauranen, and A. Haikonen, "The shift in the techno-socio-economic paradigm and regional competitiveness," 43rd Conference of European Regional Sciences Association (ERSA), 27-31 Aug 2003, Helsinki University of Technology: Lahti Center, Jyväskylä, Finland.

[21] G. Ropohl, "Philosophy of Socio-Technical Systems," Society for Philosophy and Technology. vol. 4, Virginia Tech: Blacksburg, VA, 1999.

[22] S. Reay Atkinson, Cyber-: "Envisaging New Frontiers of Possibility," UKDA Advanced Research and Assessmnent Group, unpublished, Occasional Series, 03/09, 2009.

[23] S. Reay Atkinson, I. Hassall, N.H.M. Caldwell, M. Romilly, and R. Golding, "Versatile modular system (VMS™) designs for a versatile modular fleet (VMF™)," paper presented at EAWWIV Conference, 2011, Old RN College, Greenwich, London.

[24] S. Reay Atkinson, A. Vakarau Levula, N.H.M. Caldwell, R.T. Wigand, and L. Hossain, "Signalling decision making and taking in a complex world," International Conference on Information Technology and Management Science (ICITMS 2014), May 1-2, 2014, Hong Kong: WIT Transactions on Engineering Sciences.

[25] D.S. Fadok, John Boyd, and John Warden: Air Power's Quest for Strategic Paralysis, ed. Air-University, Maxwell Air Force Base, Alabama: Air University Press, 1995.

[26] S. Reay Atkinson, "Returning Science to the Social," The Shrivenham Papers, UK Defence Academy, Number 10, July (July), 2010.

[27] M.A. Bunge, "Ten Modes of Individualism - None of Which Works - And Their Alternatives," Philosophy of the Social Sciences, 30(3), pp. 384-406, 2000.

[28] C.S. Gray, "Weapons don't make war," in Policy, Strategy and Military Technology, Editor: Lawrence, University Press: Kansas, 1993.

[29] E.N. Luttwak, The Logic of War and Peace. Revised Edition, Cambridge, MA: Harvard University Press, 2001.

[30] S. Reay Atkinson and A. Goodman, "Network strategy and decision taking," ARAG Occasional, UK Defence Academy, 11 / 08, 2008.

[31] L.A. Zadeh, "Toward a Perception-based Theory of Probabilistic Reasoning with Imprecise Probabilities," Journal of Statistical Planning and Inference, vol. 105, pp. 233-26, 2002.

[32] T.D. Clark, J.M. Larson, J.N. Mordeson, J.D. Potter, and M.J. Wierman, "Applying Fuzzy Mathematics to Formal Modelling in Comparative Politics," Studies in Fuzziness and Soft Computing, vol 225, Springer, 2008.

[33] G. Pólya, "Heuristic reasoning in the theory of numbers," reprinted in: The Random Walks of George Pólya, Ed., G.W. Alexanderson. Mathematical Association of America: Washington, DC, 1959 (2000).

[34] D. Walker, S. Reay Atkinson, and L. Hossain, "Collaboration without rules - A new perspective on stability operations," presented at IEEE Cyber Conference, 14-16 Dec, 2012, IEEE: Washington.

[35] S. Reay Atkinson, S. Feczak, A. Goodger, N.H.M. Caldwell, and L. Hossain, "Cyber-internet: a potential eco-system for innovation and adaptation," European Alliance for Innovation: Internet as Innovation Eco-System Summit and Exhibition, 4-6 Oct., 2012, EAI, Riva del Garda: Italy.

[36] D. Cartwright, Influence, Leadership, Control, in Handbook of Organizations, J.G., March, Editor, Rand McNally: Chicago, pp. 1-47, 1965.

[37] P.A. David, "Path Dependence - A Foundational Concept for Historical Social Science," Cliometrica -The Journal of Historical Economics and Econometric History, 1(2), Summer, 2007.

[38] R.A. Dahl, "The Concept of Power". Behavioral Science, 2:3, July, p. 201, 1957.

[39] L. Hossain, M. D'Eredita, and R.T. Wigand, "Towards a product process dichotomy for understanding knowledge management, sharing and transfer systems in Organizations," submitted to Information Technology and People, 2002.

[40] D.H. Wrong, "Some Problems in Defining Social Power," The American Journal of Sociology. vol. 73, No. 6, May, pp. 673-681, 1968.

[41] L.A. Zadeh, "Fuzzy Sets," in Information and Control, vol. 8(3), pp. 338-353, 1965.

[42] T.J. Ross, J.N. Booker, and W.J. Parkinson, "Fuzzy logic and probability applications: bridging the gap," Fuzzy Logic and Probability Applications. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), p. 209, 2002.

[43] R.M. Tong, Analysis of Fuzzy Control Algorithms using the Relating Matrix. PhD Thesis, in CUED, Cambridge University: Cambridge, 1976,

[44] F. Schauer, "Do people obey the law?," in Julius Stone Address, 13 March, 2014, Sydney University Faculty of Law (unpublished): Sydney University.

[45] S. Reay Atkinson, and J. Moffat, The Agile Organization, Washington: CCRP Publications, 2005.

[46] W. Waldron, "Vagueness and the Guidance of Action," Philosophical Foundations of Language and the Law. A. Marmor, and S., Soames Eds., 2011, Oxford Scholarship Online, www.oxfordscholarship.com: Oxford, retrieved March 2014.

[47] E. Mumford, "Problems, Knowledge, Solutions: solving Complex Problems," Journal of Strategic Information Systems, vol. 7, pp. 255-269, 1998.

[48] R.B. Warren and D.I. Warren., The Neighborhood Organizer's Handbook. South Bend, Ind: University of Notre Dame Press, 1977.

[49] S. Reay Atkinson, Engineering Design Adaptation Fitness in Complex Adaptive Systems. PhD Thesis, CUED EDC. Cambridge University Engineering Department: Cambridge, UK, 2012.

[50] S. Reay Atkinson, M. Mitchell, A. Wehbe, I. Wahlet, H. Ainsworth, M. Harré, and S. Sousa, "Classifying and systemising uncertainty and instability - a dynamic social network approach to risk," in RISK Conference, 2014, Brisbane, 28-30 May, Engineer Australia.

[51] S. Reay Atkinson, "Risk, Recovery and Creating Resilience - research into the Blue Mountains Fires, Oct., 2013," Productivity Commission submission on Natural Disaster Funding, B.M.C.C., S. Reay Atkinson, The Lions Club of Winmalee, NRMA Insurance, RHoK (Sydney), ENGG 3853 Risk Managment Students, Ed. S. Reay Atkinson, University of Sydney, CCSRG: Sydney. 2014.

[52] N.S.W.L.R., "Coal Cliff Collieries Pty Ltd v Sijehama Pty Ltd.," New South Wales Law Report (N.S.W.L.R.), Law: vol. 1, iss. 24, Sydney, 1991.

[53] M. Foucault, The Order of Things: An Archaeology of the Human Sciences, New York, NY: Vintage Books, 1994.

[54] S. Reay Atkinson and G. Tolhurst, "Certainty in joint venture negotiations: a case study," Commercial Law Quarterly, March-May, vol. 3, The Commercial Law Association of Australia: Sydney, pp. 3-21, 2015.

[55] S. De Sousa, I. Lopes, and E. Nunes, "On the quantification of uncertainty of performance measures," Third International Conference on Business Sustainability, November 20-22, 2013, Management, Technology and Learning for Individuals, Organisations and Society in Turbulent Environment, Ed. Goran Putnik. Póvoa de Varzim, Portugal.

[56] R.V. Rao, Decision Making in the Manufacturing Environment: using Graph Theory and Fuzzy Multiple attribute Decision Making methods, London: Springer. 2007.

[57] R. Likert, "A Technique for the Measurement of Attitudes," Archives of Psychology, vol. 140, 1932.

[58] A. Atwani, N. Tohver, and S. Reay Atkinson, "Application of Concurrent Engineering to Decision Making and Taking: delivering Sustainability Standards in Construction Projects," Atkinson SR (ed), Interim Report to Lend Lease. Sydney: FEIT and CCSRG, 2014.

[59] U.S.-N.C.T.A. "National Commission of Terrorist Attacks (N.C.T.A.)," 9/11 Commission Report, 2004, [cited 2007 June]; Available from: http://www.9-11commission.gov/report/911Report.pdf.

[60] Lord Butler, "Review of Intelligence on Weapons of Mass Destruction," Report of a Committee of Privy Counsellors, Chairman: The Rt Hon The Lord Butler of Brockwell KG GCB CVO, 2004, Ordered by the House of Commons, 14th July: London.

[61] H.I. Ansoff, "Managing Strategic Surprise by Response to Weak Signals," California Management Review, vol. XVIII, no. 2, pp. 21-33, 1975.

[62] B. Coffman, "Weak Signal Research, Part I: Introduction," 1997 [cited 2010 13 Oct]; available from: http://www.mgtaylor.com/mgtaylor/jotm/winter97/wsrintro.htm.

[63] M. Hansen, "The Search-transfer Problem: the Role of Weak Ties in Sharing Knowledge across Organization Subunits," Administrative Science Quarterly, pp. 82-111. 1999.

[64] E. Hiltunen, Weak Signals in Organizational Futures Learning. PhD Thesis. Helsinki School of Economics, vol. A-365, 2010.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
issn: 1942-2679

International Journal On Advances in Internet Technology
issn: 1942-2652

International Journal On Advances in Life Sciences
issn: 1942-2660

International Journal On Advances in Networks and Services
issn: 1942-2644

International Journal On Advances in Security
issn: 1942-2636

International Journal On Advances in Software
issn: 1942-2628

International Journal On Advances in Systems and Measurements
issn: 1942-261x

International Journal On Advances in Telecommunications
issn: 1942-2601