TAMPERE UNIVERSITY OF TECHNOLOGY

Pasi Orpana

TLS Adaptation for Virtualized Border Gateway

Master of Science Thesis

Examiner: Professor Jarmo Harju
Examiner and topic approved by the Council of the Faculty of Computing and Electrical Engineering on 12th August 2015

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
Pasi Orpana: TLS Adaptation For Virtualized Border Gateway
Master of Science Thesis, 60 pages, 2 Appendix pages
September 2015
Major: Communication Networks and Protocols
Examiner: Professor Jarmo Harju
Keywords: TLS, virtualization, border gateway, cloud computing

WiFi radio access technology is being adopted as a part of the current small cell solutions of mobile networks. Any WiFi network with Internet connectivity can be used by the mobile device to access packet-based LTE services like voice calling and instant messaging. This service traffic is strictly confidential and needs to be protected with encryption when an untrusted access, i.e., a public hotspot access, is used. This new scenario requires a new network element that terminates the encrypted service connection. For Nokia Networks this element is called the Border Gateway. At the same time the telecommunications industry is moving towards cloud computing so network elements are being virtualized to operate on virtual machines running in the cloud instead of the current embedded systems.

This thesis begins by discussing the current industry landscape and how both the WiFi small cells and the cloud-based network infrastructure are partly answering the problem of rapidly growing mobile data consumption. The focus of the thesis is studying the capabilities of the Border Gateway on the context of non-real-time service traffic encryption, with an emphasis on the virtualized platform. Also a general network security related validation is performed, as this element exists for providing security features.

The virtualized Border Gateway proved to be ten times more capable than the original embedded system, because the current x86-based processors used in the cloud offer significantly more performance than the digital signal processors in the embedded system. The fact that the Unix operating system and related software in the cloud are mostly open source, proved that security related testing only needed to be performed on the embedded platform, as there the platform software is closed-source and not already verified by the masses.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
Pasi Orpana: TLS Adaptation for Virtualized Border Gateway
Diplomityö, 60 sivua, 2 liitesivua
Syyskuu 2015
Pääaine: Tietoliikenneverkot ja protokollat
Tarkastaja: Prof. Jarmo Harju
Avainsanat: TLS, virtualisointi, border gateway, pilvipalvelut

WiFi-radiopääsyverkko on tulossa osaksi nykyisten matkapuhelinverkkojen pienten solujen ratkaisuja. Mobiililaite voi käyttää LTE-verkkojen pakettipohjaisia puheluja ja tekstiviestejä minkä tahansa Internetiin yhteydessä olevan WiFi-verkon kautta. Näistä syntyvä liikenne on täysin luottamuksellista, ja se tulee suojata salauksella, kun epäluotettavaa verkkoa, kuten julkista WiFi-verkkoa, on käytetty. Tämä uusi skenaario edellyttää uuden verkkoelementin toteuttamista, johon salatut yhteydet terminoidaan. Nokia Networksin kyseinen elementti on nimeltään Border Gateway. Samaan aikaan televiestintätoimiala on siirtymässä kohti pilvilaskentaa, jossa verkkoelementit virtualisoidaan ja ne toimivat pilvessä virtuaalikoneiden päällä nykyisten sulautettujen järjestelmien sijasta.

Tässä diplomityössä aluksi käsitellään nykyisiä toimialan näkymiä ja miten pienet WiFi-solut ja pilveen perustuva infrastuktuuri ratkaisevat nopeasti kasvavan mobiilidatan käytöstä aiheutuvia haasteita. Itse työ keskittyy tutkimaan Border Gatewayn kyvykkyyttä salata ei-reaaliaikasta liikennettä, painoittuen virtuaalisen alustan ominaisuuksien tutkimiseen. Lisäksi kyseiselle verkkoelementille suoritettiin yleinen verkon tietoturvavalidointi, johtuen siitä, että elementti on runkoverkkoon päin tulevien yhteyksien terminointipiste ja täten se on merkittävässä asemassa takaamassa runkoverkon turvallisuutta.

Virtualisoitu Border Gateway osoittautui kymmenen kertaa suorituskykyisemmäksi verrattuna alkuperäiseen sulautettuun järjestelmään, koska nykyiset x86-suoritinarkkitehtuuriin perustuvat suorittimet ovat merkittävästi suorituskykyisempiä kuin sulautetussa järjestelmässä käytetyt digitaaliset signaaliprosessorit. Pilvessä käytetty Linux-alusta perustui pitkälti avoimeen lähdekoodiin, jolle tietoturvavalidointi ei ole tarpeellinen toisin kuin sulautetun järjestelmän suljettuun lähdekoodiin perustuva alusta, jota ei ole valmiiksi validoitu suuren yleisön toimesta.

PREFACE

This Master of Science thesis work was done at Nokia Networks in Espoo, from Autumn 2014 to spring 2015.

I would like to thank my Nokia colleagues and especially my instructor Jyri Suvanen for providing the opportunity to work with such an interesting topic. Also thanks are in order for Professor Jarmo Harju who steered me through this process.

Espoo, 21st July 2015

Pasi Orpana

CONTENTS

TERMS AND DEFINITIONS

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
ATCA	Advanced Telecommunications Computing Architecture
B2BUA	Back-to-Back-User-Agent
BGW	Border Gateway
CA	Certificate Authority
CSFB	Circuit Switch FallBack
DoS	Denial of Service
e2ae	End-to-access-edge
EPC	Evolved Packet Core
ePDG	Packet Data Gateway
FQDN	Fully Qualified Domain Name
GSM	Global System for Mobile Communications
ICMP	Internet Control Message Protocol
IKE	Internet Key Exchange
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IMS-AGW	IMS Access Gateway
IPSec	IP Security Architecture
IT	Information Technology
MAC	Message Authentication Code
MSRP	Message Session Relay Protocol
NFV	Network Functions Virtualization

NRT	Non-Real-Time
OSI	Open Systems Interconnection
OTT	Over-The-Top
PKI	Public Key Infrastructure
QoS	Quality of Service
RAN	Radio Access Network
RCS	Rich Communications Suite
RTOS	Real-Time Operating System
RT	Real-Time
RTP	Real-Time Transport Protocol
SBC	Session Border Controller
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SRTP	Secure Real-Time Transport Protocol
SRVCC	Singe Radio Voice Call Continuity
SSL	Secure Sockets Layer
TAS	Telephony Application Server
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UE	User Equipment
VM	Virtual Machine
WiFi	Wireless Fidelity

1. INTRODUCTION

Possibly the largest shift in the basic structure of mobile networks infrastructure is about to happen in the near future. This change is driven by money, as is the nature of any business, but the end result for the end user is that they get to enjoy the continuous growth of cheap broadband data for years to come. The feasibility of this change is dependent on how ready the available technologies are to support this type of change. The way was paved by the IT industry which has benefited greatly from it and the telecommunications industry is now in a hurry to follow and reap the same rewards. This change is, of course, the transition to cloud-based services. Currently, the mobile network core elements are proprietary made hardware and mostly unique between manufacturers but the transition to the cloud makes it possible to separate the software and hardware businesses. The cloud provides a pool of general purpose computing power that can be used to run virtualized network element software and construct the whole core network solely of virtualized network elements.

This thesis work is the product of advances in multiple aspects of mobile communication field in general. WLAN radio access technology is standardized to being just another small cell RAN in a fully packet-based LTE network. It is possible for a public and privately managed WLAN networks to be used for making phone calls and sending text messages. Encryption is vital as these public WLAN networks are used as a part of the carrier's data path for packet-based carrier services.

This thesis focuses on Nokia Networks product called Border Gateway. This network element is responsible for terminating the encrypted connections from user end devices and route the traffic to the core network. The focus is in the part of the Border Gateway software that is responsible for the non-real-time traffic encryption like text messages and data transfer. This software was modified to operate on a x86 platform used in a cloud environment as a virtualized Border Gateway. Encryption requires the most resources in this context and the goal was partly to see how a DSP processor performs such a task and how it compares to a virtualized implementation.

Chapter 2 explains why the industry is moving towards the cloud and why we are now facing the need for encryption mechanism for user plane data. The network and cryptographic protocols that are relevant in the scope of encrypted non-real-time carrier provided service traffic are explained in Chapter 3. The Border Gateway network element is briefly explained in Chapter 4 and also the specifics of the rivalling DSP and cloud platforms. Chapter 5 explains the reasoning and methods on what was measured and why this data helps in comparing the two different platforms. Chapter 6 goes over the data and compares the results from both platforms. The goal of the last chapter is to evaluate the feasibility of a virtualized Border Gateway on the perspective of how it handles this specific encryption use case.

2. MOBILE NETWORKS EVOLUTION

We are currently moving into the fourth generation of mobile broadband networks. Long Term Evolution (LTE) is often used as a synonym for 4G networks, although this is not entirely accurate. LTE increases throughput and decreases latency of the network, but the most significant change is the migration to a packet switched all-IP network. It is the first radio access technology that is 100% packet-based, which combined with the all-IP network core, translates into greater efficiency. Eventually mobile devices could operate with a single radio module, because all the cellular services will be packet switched.

Why and where comes the need for creating a secure path for the user's traffic through the radio access network (RAN)? Is the need for encryption linked to the industry's migration to virtualized network components? This chapter provides the answers to these questions and the background for the thesis.

2.1 Telco Cloud

Mobile broadband operators are facing challenges many didn't foresee, the whole field is reacting, instead of responding to the rapid growth in mobile broadband traffic. Increasing capacity is expensive and utilizing it efficiently is problematic. Additionally, mobile networks are transitioning from being transport-centric, where the focus is on delivery of packets, to becoming service-centric, where intelligent end-to-end features deliver services efficiently and cost effectively.

The network is utilized where the users are, so a typical scenario is where the network in suburban areas is lightly utilized during business hours and highly during the evenings. This behaviour in network utilization is predictable but the unpredictable scenarios are also a problem, which currently cannot be met with traditional infrastructure. The network's resources in a particular area are static while its utilization is erratic.

Many generations of technologies are overlapping and the complexity, and all the problems that come with it, is increasing. This may be transparent to the users, or more precisely to the clients. Operators are looking for increasing revenue and they start to see the path that the IT industry as a whole has embraced.

The movement to decouple the software and hardware is not a new thing in the industry. The growing complexity is showing especially in the core network where the number of different elements has increased with every generation. The first steps to separate software from the hardware was for the industry to use a single platform for all the network elements. The Advanced Telecommunications Computing Architecture (ATCA) was specified in co-operation by more than 100 companies [1]. It provides a hardware platform for the industry as a whole to be used as an off-the-self solution. Before ATCA, different network component vendors had their own specific hardware and software. The NSN Open Core System, released in 2011, provided a single hardware platform for many different core network elements [2]. This provided flexibility for the operators and the possibility of redefining the network elements, just by switching software, to meet different traffic demands. Although, the hardware was still vendor specific to some degree, meaning that only NSN software could be used.

The next step is to decouple the business and the hardware, which is what cloud is all about and what the IT industry have used for great benefit. The move to the cloud is not as straight forward for the telco industry. The carrier grade Quality of Service (QoS) parameters and real-time computation needs of, for example, audio signal processing for a phone call are one of the challenges that the traditional cloud infrastructure cannot answer. The server hardware that cloud service providers use in their datacenters is designed for general purpose computing, which does not perform the computational tasks of signal processing very efficiently, compared to a DSP processor. Efficient architecture for a specific task consumes less processor cycles and therefore less electricity, which is a cost factor for the operators.

The change that has driven the industry towards the cloud is greatly influenced by the steady increase in general purpose, x86-based, hardware performance. The rate that DSP processors evolved is much slower and thus the gap between general purpose and signal processors is diminishing. Doing the task intended for a DSP may be slower and less efficient for a x86-based general purpose processor, but the scalability of cloud computing makes this a moot point. The way different processor architectures are developing, x86 could eventually match or surpass DSP performance in signal processing computing.

The motivation to break the link between software and hardware has mostly come from market forces. Business terms like decreasing Capital expenditure (Capex) and Operational expenditure (Opex) often surfaces when virtualization and cloud computing are mentioned. The change is also being formalized through the work on Network Functions Virtualization (NFV) by the European Telecommunications Standard Institute (ETSI), which aims to encourage international collaboration to accelerate development and deployment of interoperable solutions [3].

Telco Cloud is a long term plan for the industry. Software can be decoupled from proprietary hardware to operate on general purpose hardware through the use of virtualization layers in the software. A fully fledged Telco Cloud is automated, standardized and programmable, it defines the cloud management framework. A Telecom grade cloud environment has requirements that the existing IT clouds don't satisfy. A large number of virtual nodes with internal hierarchies and networks and a carrier grade service level agreement (SLA) of 99,999% are needs that are not met with current frameworks. Guaranteed computing resources that are always available and the possibility of soft real-time computing, meaning maximum latency guaranties for processes are also a requirement.

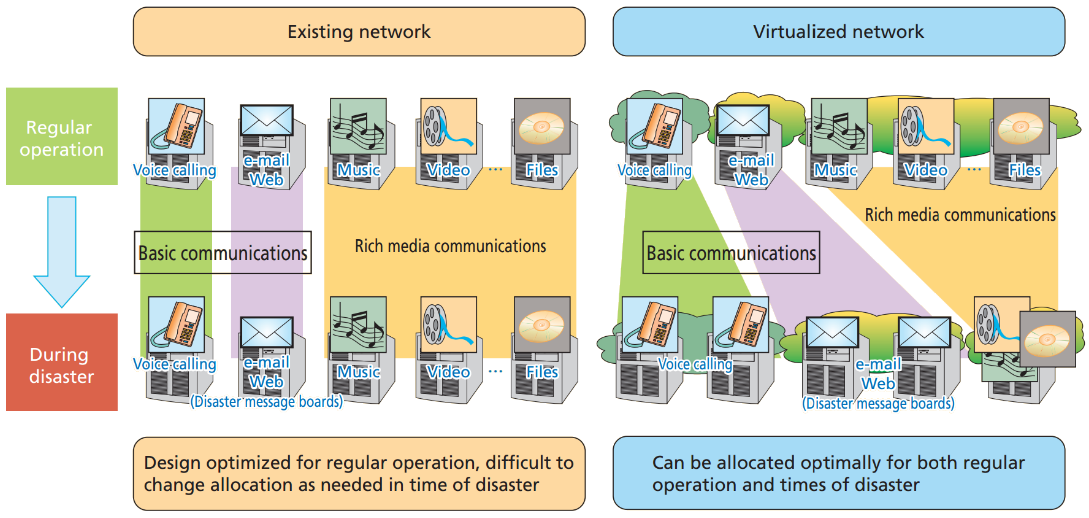
Existing network
Virtualized network
Regular operation
During disaster
Voice calling
e-mail
Web
Music
Video
Files
Basic communications
Rich media communications
(Disaster message boards)
Design optimized for regular operation, difficult to change allocation as needed in time of disaster
Can be allocated optimally for both regular operation and times of disaster

Figure 2.1: Disaster countermeasures using network virtualization [4].

How does Telco Cloud effect the end users? The possibility of accurate and near real-time automated resource allocation depending on the time of the day and the location minimizes the situations where users cannot utilize the network as they see fit due to network congestion. A more important factor is the mobile network's ability to function under unpredictable loads during disasters, natural or man made, as shown in Figure 2.1. The 2011 tsunami in Japan increased the voice call volume

50 fold from normal in the area [5]. It is safe to say that the majority of the calls did not connect and possibly lives were endangered because of this.

2.2 Core network

Operators build their networks to handle the amount of traffic they deem sufficient. The bursts of high traffic periods that are above the average should not affect SLAs and therefore capacity is scaled based on that, meaning that most of the time the valuable core network elements are utilized with relatively low loads. Core network virtualization has emerged as a key approach to maximizing network resource utilization and thereby minimizing network OPEX, meaning that operating expenditure would no longer consist of under used resources.

LTE network core is the Evolved Packet Core (EPC) and it consists of elements like the Home Subscriber Server (HSS), that can be virtualized and basically made into a just another piece of software that can be executed in a cloud environment. This can be seen in Figure 2.2, where all the EPC elements are running on different clouds and data centers, but in a more common situation all of them would be centralized in a single data center.

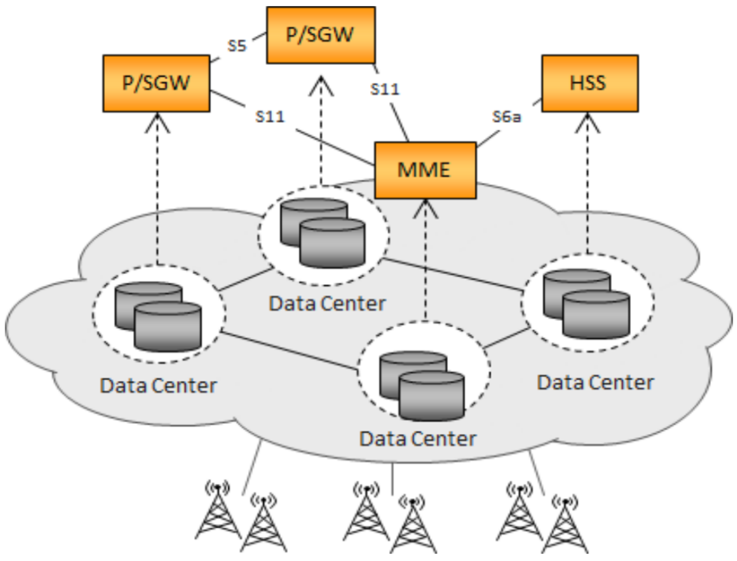

Figure 2.2: Virtualization of EPC network elements [6].

2.2.1 IP Multimedia Subsystem

IP Multimedia Subsystem (IMS) is an architecture framework defined by 3GPP [7], that defines an equipment and service agnostic system, as well as the underlying standards, like standards for security, quality-of-service, and interoperator accounting [8].

IMS stands for IP multimedia subsystem, but on today's world, the name can be deceiving. IMS used to be an IP-based subsystem inside the core network, that managed multimedia services and delivered them over to the packet switched networks. IMS is no longer a subsystem, although the name has stuck, it is the network core and everything else builds around it, as shown in Figure 2.3. IP-based networks offer superior scalability and cost efficiency compared to what circuit switched networks could offer. The rapid growth in mobile broadband traffic needs has driven the industry towards an all-IP model, which LTE networks are embracing fully. [9] IMS

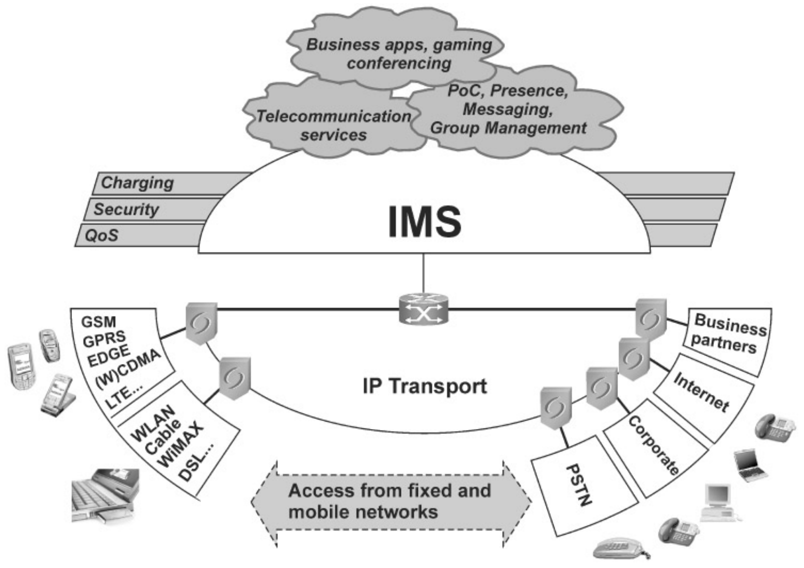
Business apps, gaming conferencing
Telecommunication services
PoC, Presence, Messaging, Group Management
Charging
Security
QoS
IMS
GSM GPRS EDGE (W)CDMA LTE...
WLAN Cable WiMAX DSL...
IP Transport
Business partners
Internet
Corporate
PSTN
Access from fixed and mobile networks

Figure 2.3: IMS domain overview, adopted from [9].

itself is a huge system with many aspects, but we are interested mainly on the new emerging access network types and the challenges they bring with the combination of new multimedia services, like VoLTE. With 3GPP's Common IMS, a wide variety of access networks are supported, most importantly WiFi.

Regarding virtualization, IMS core is also a prime candidate for it because of its compute-oriented nature, so it is suited to hardware that can leverage memory and CPU performance and benefit from the rapid upgrade cycles in the commercial server market [10]. For example Telephony Application Server (TAS) is one the key component of IMS core and also for handling the packet-based services, which are discussed in Subsection 2.3.

2.2.2 Heterogeneous networks

The number of users and services that utilize mobile broadband access is increasing at such a rate that mobile operators are struggling to keep up with the demand. Estimation done by The Strategy Analytics showed that mobile data traffic grew by 100% in the year 2012 [11]. Data traffic was forecasted to increase by 400% by the year 2017. In 2013, the number of mobile-connected devices exceeded the world population and by 2017 there will be about 1.4 mobile devices per capita [12]. Figure 2.4 illustrates the data growth forecast up to 2017.

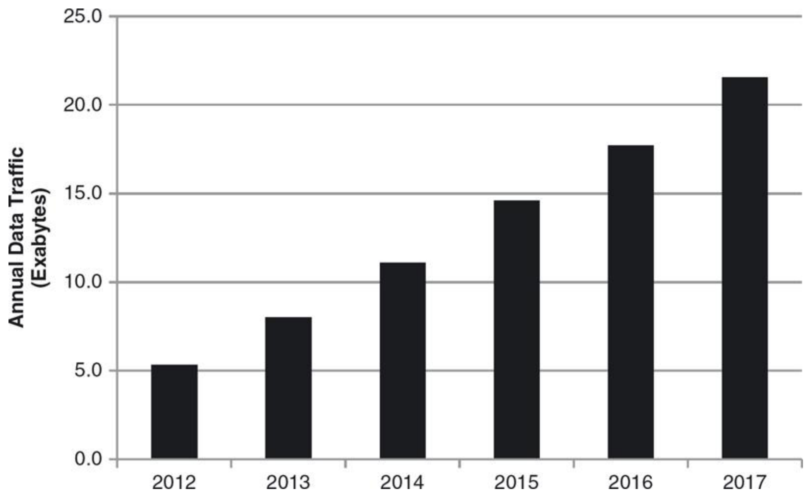

Figure 2.4: Growth forecast in annual mobile data traffic [11].

The way to increase capacity is to increase spectral efficiency, spectrum amount, and network density. Little improvement can be gained by working on the first two because current networks are already operating at near optimal regarding these issues. Network density cannot be increased by using the same radio access technology as overlapping cells start interfering and causing more harm than good.

Places where the network utilization is highest are usually centralized to certain localized indoor areas such as shopping malls. These hotspots can use most of the resources of the whole macrocell that is supposed to serve a large area. Traditionally the way to increase network capacity was to set up more macrocell base stations. At the current rate macrocell deployments are increasing capacity 30% less than the demand for data [13].

Smaller cells inside the macrocells can be used to offload traffic from the macrocell and improve frequency reuse. The term small cells encompass indoor femtocells, outdoor picocells or a compact base station microcell. Additionally, small cells can offer increased traffic capacity compared to the macrocell, as they can adapt better to traffic variations by employing dynamic interference management techniques [13].

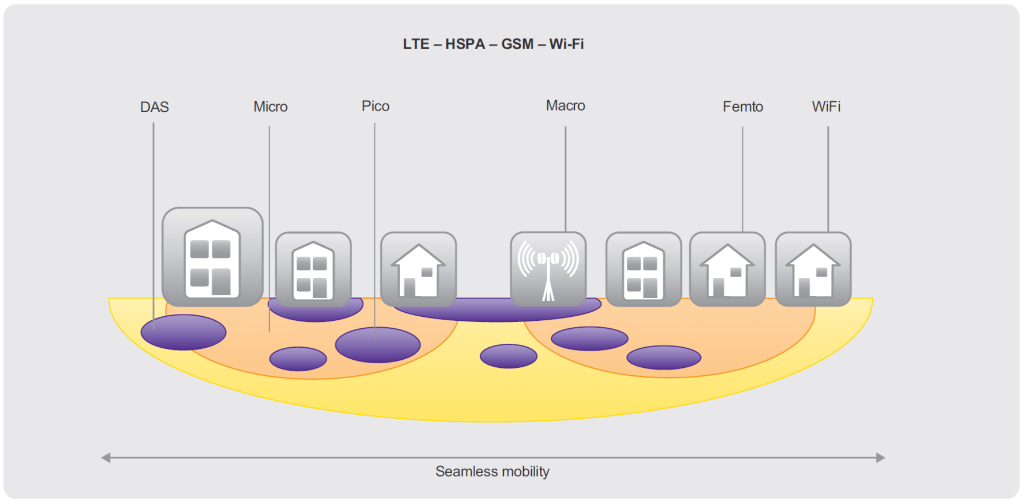

Figure 2.5: Different RAN technologies overlapping forming a heterogenous network [14].

The different kind of small cells have low transmit power and coverage depending on the type. There's no clear definition for the cells but femtocells usually have a range of 10 meters, picocell 200 meters and microcells up to two kilometers. Networks that utilize both macro and small cells are referred to as Heterogeneous networks or HetNets. Figure 2.5 illustrates the overlapping of different sized cells. HetNet with hundreds or thousands of small cells requires extra control traffic that stresses the transport and core networks so that the complex network operates seamlessly. HetNets can be complex networks so many of the conventional controlling operations have been automated so that the cell deployment and operation would be as simple as possible.

Small cells introduce security related challenges that were not present with macrocells. Small cell basestations can be situated in varying locations so physical security can be non-existent or the location is managed by a 3rd party so physical tampering needs to be anticipated. In most cases inherently insecure public Internet connection is used for access to the core network and therefore the traffic needs to be encrypted.

Operators are trying to increase capacity while reducing the overall costs and small cells are providing both. Macrocell basestations are less energy efficient than their small cell counterparts as their power amplifiers require separate cooling. Off-loading traffic to small cells have been found to reduce the overall energy consumption by

25% to 30% [13].

The industry has adopted small cells as an integral part of the modern wireless broadband network as femtocell deployments made up 56% of the global base station deployments [13]. Small cells and heterogeneous networks play an integral role in the global mobile broadband infrastructure.

2.2.3 WiFi offloading

WiFi has become a standard feature in virtually every mobile device. The unlicensed spectrum and low-cost hardware creates great value for users and operators. WiFi networks are also prevalent as they are commonly used in public hotspots, private networks and business environments. 70% of mobile broadband traffic occurs indoors so the majority of the network load is generated in potential WiFi coverage areas [15].

Offloading traffic from the congested mobile network to a WiFi small cell has many benefits. Unlicensed spectrum makes it basically "free" for operators to exploit. Also the spectrum is different compared to existing small cell networks so no interference among overlapping cells. Close proximity to the base station require lower transmit powers so battery usage is smaller compared to using a mobile RAN.

Small cell WiFi networks are not always managed by the operator itself, they can be managed by 3rd party providers. The goal is to use whatever WiFi network is available and in most cases they are 3rd party networks. One additional benefit from this is that the backhaul connection is also maintained by the provider so the operator don't need to bear the expenses.

The base stations can also be located in places, where physical security cannot be guaranteed. These small cells provide security of varying strengths, so on the operator's point of view, they can be considered untrusted networks. It is therefore a requirement for operators to have media plane security, which provides uniform protection against eavesdropping and undetected modification across access networks.

3GPP specification TS 23.402 describes the integration of trusted and untrusted IP access networks into 3GPP systems [16]. The intent of 3GPP-WLAN interworking is to extend 3GPP services and functionality to the WLAN access environment. It provides the packet switched bearer services to be used over a WLAN connection. The standard accepts that a WLAN is as valid an access network as any other 3GPP

RAN. Untrusted access means that encryption is required so a secure path needs to be formed over the untrusted network. IPsec is the standard solution to create a secure tunnel to the ePDG (Evolved Packet Data Gateway) which terminates the tunnel and operation is as usual after that point, as illustrated in Figure 2.6. This needs authentication using the SIM card data and therefore needs multiple mechanisms that devices can transparently use WLAN access.

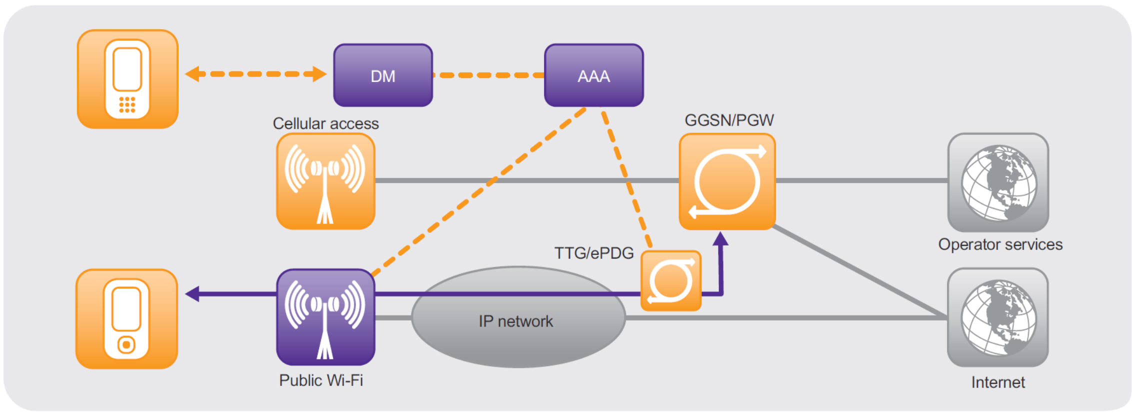

Figure 2.6: Path from an untrusted Wi-Fi RAN to operators network, adopted from [14].

Seamless WLAN offloading in mobile networks is coming but is not yet supported as a standard way of every day usage. Before it is implemented out as a feature, the existing WLAN networks are already preferred by the devices to be used for general, non-bearer traffic, if they are available. This is basically manual offloading done by the users, if they connect the device to an locally available WLAN network. The end goal of WLAN offloading is to take the pressure out of the macro networks when possible to offload everything, including packet-based bearer service traffic, which are discussed in Section 2.3.

WLAN offloading is at the focal point of this thesis as it introduces the concept of untrusted networks to be used to access core IMS network. Operator provided services situate in the IMS core and can be used over untrusted networks so service traffic is encrypted with security methods like the Transport Layer Security (TLS). This level of encryption is not employed if the used access is deemed trusted, for example if the mechanisms are in place and the device forms the IPsec tunnel over the untrusted network, then it is considered trusted access and additional encryption is not needed.

2.3 Services

Making voice calls and sending text messages are the traditional cellular mobile services that are supported by all compliant devices. No matter the country or the model of the device, these services are natively supported. The first 2G network was launched in 1991 called Global System for Mobile Communications (GSM) which has circuit switched voice calls and text messages [17]. All calls and text messages sent in a 3G, and with some exceptions in 4G, networks still use a circuit switched network, which exists only for these services. This means that there are overlapping legacy network technologies in the access and core network, which increases complexity. The overlap of circuit and packet switched domains is seen in Figure 2.7.

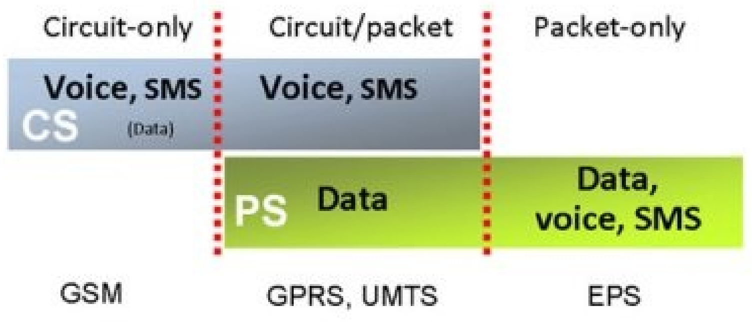

Figure 2.7: Circuit and packet domains, adopted from [18].

The majority of mobile operators revenue comes from voice and SMS services, which is a problem because the revenue from these sources is dwindling. Data traffic is increasing rapidly partly because voice and Instant Messaging (IM) needs are met with 3rd party applications. Flat-rate data subscription plans are popular among operators but problematic because they start to eat away the profits from other sources. The rapid adoption of smartphones and availability of affordable broadband connectivity has provided consumers access to a wide variety of communication services which go beyond the traditional services of voice and text messaging provided by mobile operators. The first to fulfil the potential were Over-The-Top (OTT) applications, made by 3rd party developers.

The rise of these OTT services has fragmented the communications landscape, e.g., it is not possible to make a VoIP call from one application to another. Although, using OTT services is basically free if a flat-rate data subscription is presumed. The reason why this has not replaced traditional calling completely, is that OTT communication applications are closed ecosystems that are not interoperable and cross-platform support is rare. Consequently, OTT services are dependent on mobile broadband coverage and the willingness of subscribers to use a service that lacks quality, security, seamless mobility and flexibility. This fragmentation has provided an ideal opportunity for mobile operators to mend the decline in revenues.

There are two types of communication services, Non-Real-Time (NRT) and Real-Time (RT). Voice communication is RT and instant messaging, or SMS, a NRT service. The fundamental differences of these service types impose very different requirements and therefore they are implemented with differing technologies, which are discussed in the following subsections.

2.3.1 VoLTE

LTE is an all-IP technology so on the long term, all the cellular services will be packet switched, as circuit switched networks will not be maintained indefinitely. Voice over LTE (VoLTE) is the prevailed technology that will be used for voice calls in 4G networks. For VoLTE to differentiate itself from OTT services, operators need VoLTE to provide carrier grade capabilities and a guaranteed Quality of Service (QoS). This means VoLTE must deliver the same level of quality and reliability that has been the norm in circuit switched services, which can be challenging at first when noting the fact that CS services have matured over two decades.

The QoS parameters of a voice call are strict but straight forward to manage in a circuit switched network, where a sufficient portion of the capacity is allocated for the ongoing call. The challenges are different when moving to a packet switched network. IP networks operate on a best effort model. Real-time applications like voice calls need a certain promise of latency and also need to work in heavily congested network. To monitor and manage the QoS of any VoLTE session, operators must record and co-relate these measurements per subscriber, location and device type in all layers including the network, transport, IMS and application layers [19].

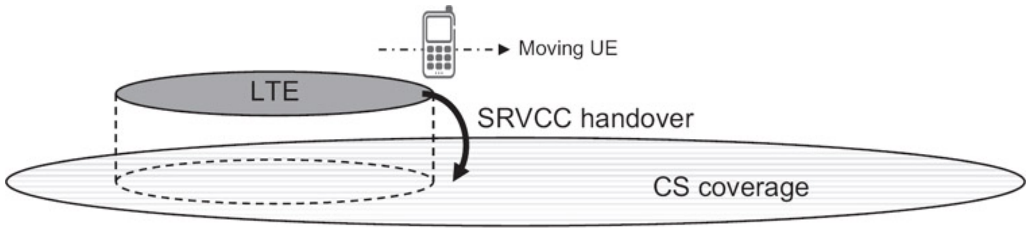

Figure 2.8: SRVCC handover scenario [20].

LTE and VoLTE market penetration is gradual so different generation of networks will be maintained parallel for a long time. During the transitioning period, where 4G networks are not fully covered everywhere, there needs to be a backup plan for voice calls. Before VoLTE is supported on the 4G network, calls will use Circuit Switch FallBack (CSFB), meaning that the underlying 2G or 3G network will be used. After VoLTE deployment, if one participant of the VoLTE call leaves the LTE network coverage area, as illustrated in Figure 2.8, the call needs to be switched over to the new network. This handover mechanism is called Single radio Voice Call continuity (SRVCC). The call will be seamlessly handed over to a circuit switched bearer. This is one opportunity where VoLTE has the ability to differentiate itself from OTT services, because no such mechanisms exist there.

VoLTE is expected to go beyond just voice. Video calling, file sharing and instant messaging services will complement VoLTE for it to compete and hopefully surpass the OTT provided services. These complementing services are implemented using Rich Communications Services which is discussed in the next section.

2.3.2 Rich Communications Services

Rich Communications Services is an industry driven initiative run by GSMA. It brings together all the decision makers in the telecommunications industry to create an "interoperable, convergent, access-technology-independent rich communications experience to end-users" [20]. It doesn't define new standards, but brings together already defined services into profiles based on global standards of the IP Multimedia Subsystem.

VoLTE can be seen as a foundation service for RCS as they share the same IMS investment and leverage the same IMS capabilities. The RCS ecosystem can also work as an incentive for operators to shift existing 2G/3G voice services to an IMS architecture to tap to the new emerging value added by RCS.

RCS was launched by GSMA in 2008 to provide Rich Media services that made use of the plentiful mobile broadband data that was becoming more prevalent. There has since been many national trials and tests but RCS needed to be supported by every device and platform to achieve what it was set out to do, which was to provide out of the box support for UE devices for packet-based messaging. Commercial launch has taken longer than expected so OTT-services like Skype, Google Talk and many more were deploying faster.

With VoLTE and RCS operators can provide the same services that OTT-services can and also improve upon the experience. The necessity of RCS lies with two main points: Global interoperability between socially driven services and moving from platform-centric to subscriber-centric social networking [20]. If the world adopts RCS as is envisioned by GSMA, the way how social information is shared will change to something different.

The goal is to make RCS a native client to every device, so no additional installation is needed as it is built in the networks and devices. OEMs (original equipment manufacturers) are governing the ecosystem, instead of a 3rd party developers. All service capabilities are available by default, which makes it much easier for subscribers to find and select attractive new services, significantly lowering the threshold for adopting them.

RCS-e initiative was launched to lower the entry barrier for RCS by allowing operators a way to use RCS without implementing the full service profiles. The RCS-e subset of services are carefully selected based on those with highest consumer demand. Chat, file sharing and the ability to share live video or images during a call are the RCS-e specified services. Social profile sharing is an optional service that can be supported depending on the operator.

VoIP applications, including VoLTE, operate over UDP to meet the real-time needs of voice calls. The only RCS-e real-time service is video streaming which uses Real-time Transport Protocol (RTP) over UDP. Chat and file transfer naturally use TCP as the transport protocol. The application level protocol that most of the RCS services use is Message Session Relay Protocol (MSRP) which will be studied in more detail in Chapter 3.

3. PROTOCOLS

The BGW uses many different kinds of transmission and signalling protocols in its operation. This section aims to provide an overlook, in sufficient detail, on the relevant network protocols that are in the scope of this thesis. When creating a secure path trough an untrusted, 3rd party managed, access network, encryption is needed to bring the security aspect to a carrier grade level. Cryptographic protocols are needed to create a secure path of communications trough an untrusted network. Two UE devices communicating using VoLTE/RCS require end-to-access-edge (e2ae) security.

3.1 Definition for Security

Before going deeper in to the inner workings of TLS protocol, it's important to give a clear definition for the words like secure or security on the context of IT network communication. Such words are easily thrown around without clear meaning or understanding.

The Dolev-Yao model is a formal model that is used to prove properties of interactive cryptographic protocols. The algebraic model is not relevant on our context but the model defines the network and the adversary. The network consists of abstract machines that exchange messages and the adversary, or attacker, who carries the messages [21]. The attacker has complete control over the network but cannot compromise the machines themselves. The TLS and SSL protocols have been designed to be secure in the Dolev-Yao model [22].

In general, a security definition must answer the following points [22]:

1. What are the capabilities of the attacker? For example computing resources, time, types of attacks.
2. What is the task that the attacker must accomplish in order to be successful,

i.e., figuring the secret key used for encryption.

For a strong definition, we assume that the attacker is as powerful as possible and tries to achieve the simplest of tasks. When the attacker fails to get any beneficial information from the messages that are going trough the network, even when he has complete control of the network, time and resources, then we can say that the machines can securely exchange messages.

3.2 TLS - Transport Layer Security

Transport Layer Security (TLS) is a set of cryptographic protocols that can provide a secure communication path trough an insecure network. The largest such networks is the Internet and TLS is the standard for websites to support. In a recent survey, it showed that 99.2% of the most popular websites on the Internet supported v1.0 of the TLS protocol [23].

Secure Sockets Layer (SSL) is the predecessor of TLS. The first public version of SSL was v2.0 but it is considered insecure and majority of websites don't support it [23]. SSL v3.0 and TLS v1.0 are supported by almost every website. Newer versions of TLS are v1.1 and v1.2 but, at the time of writing, only about one third of the most popular websites had support for the latest versions.

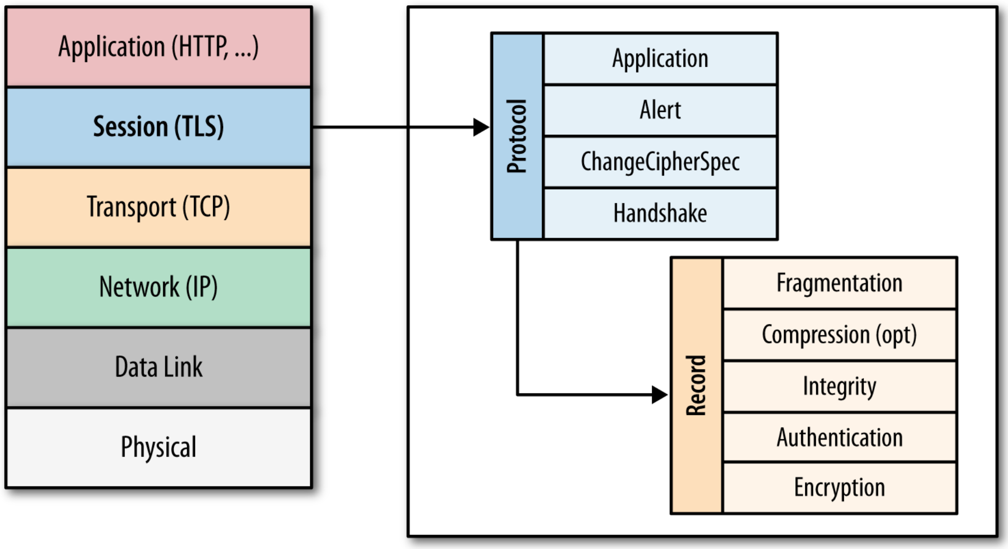

Figure 3.1: Transport Layer Security and protocol stack. [24].

As the name suggests, TLS provides security for the transport layer, meaning that it operates between transport and application layers in the OSI model, as shown in the

Figure 3.1. Applications work on top of TLS and can do so without any knowledge of the underlying protocols, meaning that it is application protocol independent. The TLS protocol provides everything from authentication and key management to encryption and integrity checking. The protocol itself has two different phases of operation: setting up the connection and steady-state communication.

In this chapter, we are going trough the TLS protocol operation from start to finish on a general level, with an emphasis on resource usage on different parts in the protocol. We are interested how much computing resources do the complex cryptographic calculations require and how these requirements might reflect on the actual usage in different CPU architectures.

3.2.1 Protocol details

As stated before, TLS protocol can be divided in to two components: the TLS Record protocol responsible for data transfer, and the TLS Handshake protocol. As TLS operates over TCP/IP, the three-way TCP handshake must be performed before anything regarding TLS can be done. TLS is an asymmetrical protocol, meaning that it differentiates between the client and the server [24].

Once the TCP connection has been established, the TLS handshake can begin. The TLS Handshake Protocol is responsible for authenticating the server and the client to each other and to negotiate the symmetric encryption key. Usually only the client is interested about the servers credentials, but it is possible for the server to ask for client certificate, although this option is rarely used. Figure 3.2 shows a high-level sequence diagram of the handshake protocol.

The handshake goes as follows:

1. The client sends a "Client hello" massage, along with clients random value and supported cipher suites.
2. Server answers with "Server hello", with random value, server certificate, session ID and the selected cipher suite.
3. Client generates a random Pre-master Secret and encrypts it with the servers public key that was received in the certificate.
4. Server and client generates the Master secret and the session keys from the Pre-master secret.

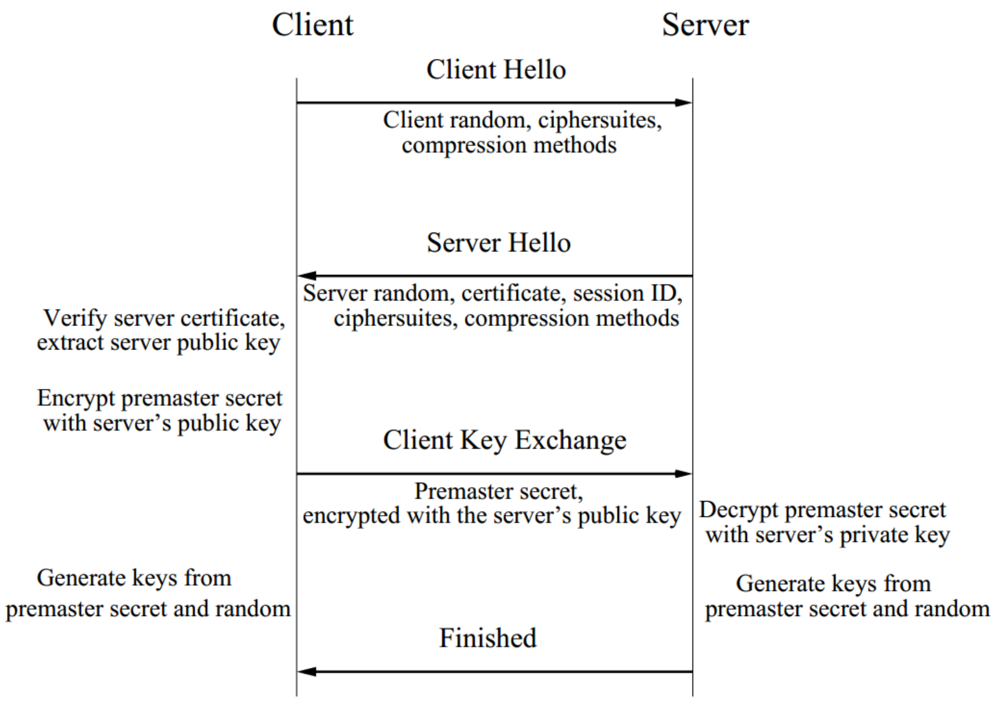
Client
Server
Client Hello
Client random, ciphersuites, compression methods
Server Hello
Server random, certificate, session ID, ciphersuites, compression methods
Verify server certificate, extract server public key
Encrypt premaster secret with server's public key
Client Key Exchange
Premaster secret, encrypted with the server's public key
Decrypt premaster secret with server's private key
Generate keys from premaster secret and random
Generate keys from premaster secret and random
Finished

Figure 3.2: New TLS connection handshake [25].

5. After this both parties have agreed on the session keys and they can transfer data securely using this key for symmetric encryption.

The handshake procedure is the most resources demanding operation of the TLS protocol. Because of this, session caching is an option to reduce unnecessary CPU cycles. If the client reconnects, only the session ID needs to sent so the server can use the corresponding cached Master secret to generate the session key.

After the connection is setup, data can be transferred, as illustrated in Figure 3.3. First, if the used block cipher is AES, the data is broken in to 16kB blocks and compressed, if it was negotiated during the handshake. A Message Authentication Code (MAC) is calculated from the packet and added at the end of the packet, along with a sequence number. The compiled packet is then encrypted using the session key with the negotiated encryption method.

Cipher suites are a set of cryptographic primitives that individually perform a very specific task, but putting them together as a suite, they form the cryptographic system. The used suite for the session is first suggested by the client during the first message in the handshake. The client sends a list of supported suites and the server selects the preferred suite, which is usually the strongest. The suite needs to

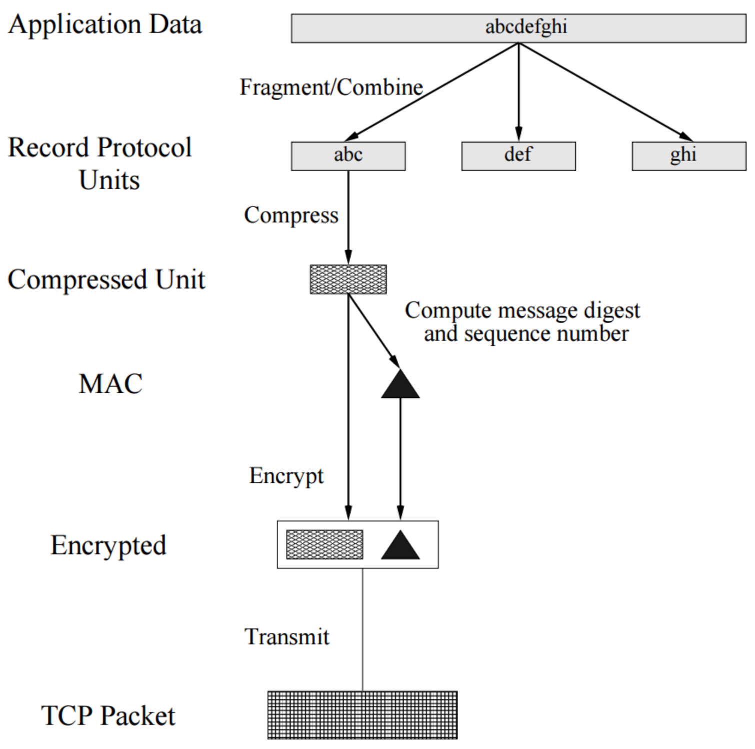

Figure 3.3: TLS processing steps during data exchange [25].

be supported by both peers, but the server selects the one it wants and it can also abort the handshake, if it doesn't accept any of suggested suites. The client can suggest any suite, even the weak ones.

There is a balance between security and how much computing resources the suites need. Suites, that provide weaker security are usually faster and vice versa. For example, using a public key size of 2048 bit compared to 1024 bit key, provides better security but is more consuming to compute.

Ciphers suites define cryptographic primitives for three different functions: Authentication, encryption and message authentication. Authentication is done with asymmetric encryption, which means the use of public and private keys. Encryption is done with symmetric encryption primitives as message authentication is done with hash functions. There are many different combinations of primitives. The cipher suite format is for example: `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`. First is the authentication method. In this case the key exchange method is ECDHE and with RSA signing. Data encryption method is AES with 128bit key and message signing is done with cipher block-chaining a Secure Hash Algorithm (CBC-SHA). The TLS 1.0 specification mandates, that a TLS 1.0-compliant application must implement cipher suite `TLS-DHE_DSS_WITH_3DES-EDE_CBC_SHA`. This is the only one that must be supported by applications, all the others are optional [22].

3.2.2 Certificates - X.509

TLS and SSL use Public Key certificates, but the specification of the certificates is outside the scope of TLS and SSL specification. The used standard for Public Key Infrastructure (PKI) is the X.509 system. It defines the format for the public key certificates, certificate revocation list and attribute certificates, among other things. The same format for certificates is used by many other cryptographic protocols, such as IPsec and IKE. RFC specification describes public certificates as a special case of certificates that binds its owner to a public key value. As such, it is a digitally signed data structure that attests to the true ownership of a public key [22].

During a TLS handshake the server sends its certificate that holds its public key which is digitally signed by a Certificate Authority (CA). Without the CA signature, it cannot be verified, that the public key belongs to the entity that is specified by the certificate. The client encrypts a message using the public key and knows that the only one who can decrypt the message is the holder of the corresponding secret key. A trusted intermediary is needed to have a trust anchor that can be trusted completely. When a CA signs a certificate with its digital signature, then it can be trusted that this public key corresponds to the owner with as much certainty as the trust you place on that CA.

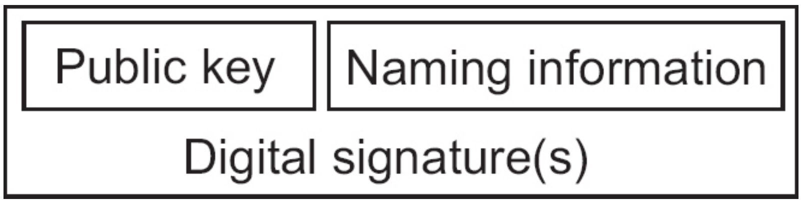

Figure 3.4: A public key certificate comprising three pieces of information. [22].

A certificate holds three key pieces of information. The public key, naming information and one or more digital signatures, as shown in Figure 3.4. The naming information specifies the owners name, and more importantly, the network address of the public keys owner. The digital signature is made by the CA by calculating a hash value from the key and naming information. The calculated hash is then "signed" by applying encryption using the CAs private key.

The validity of the certificate can be made sure by calculating the hash value from the certificate and decrypting the signature using the CAs public key. The hash values are the same, if the certificate is the same from which the CA calculated the signature.

Validating the digital signature requires to have the CA public key. Every application or system that uses X.509 certificates need to have all the root CA public keys locally on file. These are certificates that are self-signed by the root CAs. They are the trust anchors that can be depended upon. There are many dozens of established root CAs whose certificates are included for example in web browsers.

Self-signing enables anyone to create a certificate and sign it by themselves. There is no difference in the viability of the certificate if it is self-signed or signed by a CA. The only difference is that for a CA to sign a certificate, they validate the business and domain in question for whose certificate they sign. That is the product they are selling, trust. In the case of web browsers, they always expect for certificates to be signed by an approved CA, a self-signed server certificate will flag the server as potentially risky.

3.2.3 Performance analysis

The BGW (Border Gateway) will operate on two different platforms, the DSP platform and the virtualized x86-based platform. As complex as cryptographic calculations are, the computational costs are dwarfed by the performance of modern processors. For example, Google started to use TLS encryption for every connection to Gmail and the encryption takes around 1% of the servers resources [17]. As the DSP processor architecture is made for signal processing and x86 is for general purpose computation, the capabilities of cryptographic calculations on both platforms need to be investigated whether they have any possible effects on the normal operation of the BGW.

As noted earlier, asymmetric encryption is much more computationally expensive than symmetric encryption. During a TLS session asymmetric encryption is used during the handshake and symmetric for data transfer. The overall effects of the encryption methods depends on different factors, biggest of them being the handshake procedure.

RSA is one of the most used public-key cryptosystems, so using that as a baseline comparison is convenient. In public-key cryptography, modular multiplications are the most expensive operations. Multiplying large numbers and calculating their modulus are the basic operations in RSA. The way RSA works, the maximum message size is the same as key size, including a minimum of 11 bytes of padding. With a 1024-bit key the encrypted message can only be 128 bytes long, with 2048-bit key it's 256 bytes. There is no standardized way of splitting data into blocks with RSA. It is not meant to function like a block cipher. By comparison, encrypting 256 bytes worth of data with RSA takes hundreds of times longer, compared to a symmetric encryption method.

One of the most common symmetric encryption algorithms is AES. Since 2008, Intel processors have had an instruction set called AES-NI. This enables the processor, on the hardware level, for optimized AES algorithm calculation. Intel states, that for non-parallel modes of operation, AES-NI can provide two to three fold gain in performance [26]. This can be tested with OpenSSL toolkit `speed` command, which is used to test the performance of cryptographic algorithms.

```
Command A = openssl speed -elapsed -evp aes-128-cbc
Command B = OPENSSL_ia32cap="~0x200000200000000" openssl speed \
            -elapsed -evp aes-128-cbc

Block:  16 bytes     64 bytes     256 bytes    1024 bytes   8192 bytes
----------------------------------------------------------------------
A       542554.85k   573966.78k   586127.19k   585141.93k   587923.4
B       251690.11k   295278.08k   304876.29k   311101.78k   309996.20k
```

The test was run on a virtual machine (VM) with Intel Xeon CPU E5-2665 @ 2.40GHz. The 'numbers' are in 1000s of bytes per second processed. Command A is with AES-NI enabled and in B while disabled. No matter the block size, AES-NI acceleration provides the performance Intel claims, around two times faster when compared to the non-accelerated case. The DSP processor we are testing against is a Texas Instruments TMS320TCI6486 [27]. Core speed is 625Mhz and as there is no instruction set for AES, only one result:

```
Block:  16 bytes     64 bytes     256 bytes    1024 bytes   8192 bytes
----------------------------------------------------------------------
A       27962.03k    41943.04k    41943.04k    41943.04k    41943.04k
```

DSP processors are, as the name suggests, made mainly for signal processing, meaning that the processor instruction sets are related to that field, for example audio processing algorithms. There are no instruction sets that optimize cryptographic algorithms in the current DSPs that are used by the telecommunications industry. This is a big advantage for the virtual platform, as seen in these tests with more than ten times the performance difference between the platforms. Although, a new generation of DSP processors from Texas Instruments are coming in the near future that have instruction sets for cryptographic algorithms, but the actual performance benefits are still up for speculation [28].

The loads that the different algorithms introduce to the system depend on the use profile. Every new connection requires the expensive handshake to be performed. If the user base operates in a way that the connections lasts long and the users don't change very often, the computational costs, in the long run, comes from the symmetric encryption. In a case where users connect to the network, performing the handshake, transfer a small amount of data and then disconnect, it raises the impact of the handshake procedure, because that is what is done most of the time. The use profile of the connections can be informatively speculated, but no system level design decision can nor should be made upon this speculation. It is safe to assume that a large number of diverse users generate a dynamic traffic pattern, while they are using different services like sending a single text message or making long lasting video calls.

TLS has multiple mechanisms for combating the impacts of the handshake. To avoid a full handshake, the client can request a abbreviated handshake. The server will send a session ID in the "Server Hello" reply, as shown in Figure 3.2. In order for this to work, the client and the server need to store the session IDs for later use. The client can then resume the previous connection by attaching the session ID in the "Client Hello" message. The server can then find the session information in the cache and resume the connection. The benefit from this is to skip the more costly part of the handshake, the public-key cryptographic operations. The previously negotiated session key is possessed by both parties, so they can start using symmetric encryption.

Another mechanism is the usage of session tickets, detailed in RFC 5246 [29]. The server can include a "New Session Ticket" during the last full handshake, this is not represented in Figure 3.2. The ticket contains the complete session state, including the master secret, the cipher suite used and etc. This ticket is encrypted and authenticated by the server, with a encryption key only known to it, using

AES128-CBC-SHA256. The client can then send this ticket during the "Client Hello" message and the session can be resumed.

There are pros and cons for using either of the previous methods. Caching the session IDs takes memory from the server and sharing the cache between load-balancing hosts can prove difficult. It is, however supported natively by TLS, unlike session tickets. The ticket mechanism is an optional extension to TLS and is therefore not as widespread as support for session identifiers. The main benefit is that servers don't need to maintain a session cache, the tickets are held by the clients and all the information to resume a session is encrypted within. The used encryption method for the tickets, AES128-CBC-SHA256, has faced criticism. It is defined by the RFC and cannot be changed by the user. This means, that the ticket can be encrypted less securely, than the resumed TLS connection.

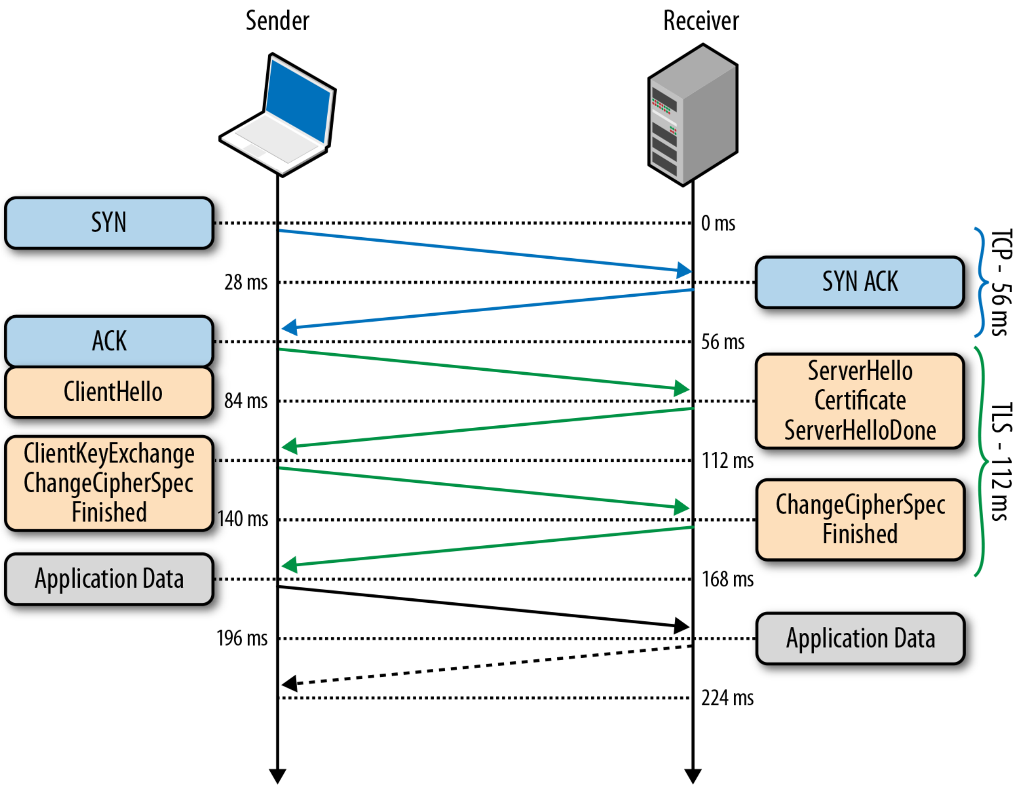

Figure 3.5: TLS handshake protocol with latency. [17].

TLS, inadvertently, adds latency. Every connection starts with the TCP three-way-handshake, which causes a delay of one full round trip time, as shown in the Figure 3.5, where the one-way delay is selected as 28 milliseconds. As TCP takes one RTT to initialize, TLS takes two RTT, so the add in delay is significant. The add in latency from using TLS can have an effect on the system and therefore should not be overlooked.

3.2.4 OpenSSL

It is possible to create an implementation of TLS protocol by following RFC 5246 and starting to write code but creating a robust implementation of an encryption protocol is something that even professionals haven't fully managed to do. Encryption is rarely done without a good reason and it only takes one flaw in the code to compromise the whole thing. There are many encryption frameworks freely available and OpenSSL is the most widely used. It is included in many Linux distributions and can be considered the standard implementation.

OpenSSL encompasses more than the name may suggest. It is a set of many different cryptographic algorithm implementations and offers an application programming interface (API) so that it can be used by programmers. All versions of SSL and TLS protocol implementations are included which require many different cryptographic algorithms to work. These basic cryptographic functions are also offered for use over the API, for example calculating a SHA-1 hash value for a given input data.

The project has been ongoing for about fifteen years now and is still compatible with the software and hardware from the previous millennia. The code base is so littered with legacy implementations that they obfuscate the code and make it very unpleasant to work with. This is the main problem of the project that is supposed to use the resources of the open-source community but the reality is that most volunteers are turned off by the poorly managed code.

The greatest attribute of OpenSSL used to be the fact that its source code is open for public and anyone can look for bugs and vulnerabilities. But who would use an encryption library whose source code is not available for viewing, as this would prevent the verification of the encryption process? There would need to be complete trust to the maker of the library as without it there are no assurances of the soundness of the encryption. If this privately developed library would become as prevalent as OpenSSL, then there would be an authority in the world who could have implemented a way around the encryption and could potentially decrypt any encrypted message. These kind of doubts will always exists and therefore the only way to have a globally used encryption library is to make it open-source.

The open-source nature of OpenSSL also used to be the reason why it is so secure. If anyone in the world can inspect the code and correct bugs, how can there be any bugs left? This false sense of security is rapidly changing with the recently discovered vulnerabilities in OpenSSL. At the time of writing the so called Heartbleed bug was discovered that used a vulnerability in the TLS heartbeat extension. When exploited the heartbeat responses contains up to 64kb of data from the servers internal memory. So it is possible to get private information from the servers memory, including passwords, master keys, etc. When the discovery was made public, around 17% of secure web servers in the Internet were vulnerable[30].

Heartbleed was such a serious flaw that another development team started their own project called LibreSSL. Recent versions of OpenSSL contains about 388 000 lines of code. The LibreSSL team removed 90 000 lines of code in the first month without effecting normal operation. It is a promising project but it was primarily developed for OpenBSD operating system and has limited support for other platforms so replacing OpenSSL with it is not yet fully possible.

3.3 Multimedia Communication protocols

Delivering multimedia content in the Internet is enabled by a multitude of IP-based application-level protocols. These protocols form the core of multimedia communication systems architecture and no system can exist without them [31]. They work in unison for content delivery and can be categorized into two groups, as illustrated in Figure 3.6.

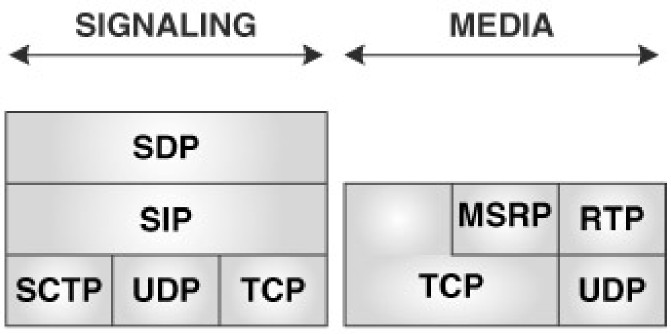

Figure 3.6: Multimedia communication protocol architecture [31].

Session Initiation Protocol (SIP) is the core signalling protocol. It is responsible for all aspects of session management and therefore used for establishment, modification and termination of the session. Session Description Protocol (SDP) messages contain generic information about the multimedia sessions and they are transported inside the body of SIP messages. This separation of management and description into independent functions is a powerful concept. It allows both mechanism to evolve without effecting each other and therefore making them forward compatible for future developments [31].

The actual content delivery is the responsibility of media transport protocols which are:

- Real-time Transport Protocol (RTP)
- Message Session Relay Protocol (MSRP)

RTP, as the name suggests, is designed to handle real-time media and to cope with the delay variations that are inevitably introduced by the nature of the underlying IP-network. RTP over UDP is the used transport protocol in VoLTE calls. For connections over an untrusted access network, Secure Real-time Protocol (SRTP) can be used for encryption.

In the scope of this thesis, we are more interested about the Message Session Relay Protocol (MSRP) as it is the used media transport protocol in the RCS context. Operating over TCP, it is used for non-real-time features like instant-messaging, photo sharing and file transfer. Even though MSRP is the top level protocol that is being forwarded through the BGW in the RCS use case context, the connection isn't terminated on the BGW and therefore it is just bits that are routed through the BGW, so a brief overlook on the details of MSRP is sufficient.

As previously stated, the sessions are initiated using SIP with SDP and the actual content delivery is done with MSRP messaging. MSRP is a relatively simple protocol with similar syntax as other IETF-based protocols such as SIP and HTTP. Messages are either requests or responses, but not every request has a corresponding response. Request message types are SEND, REPORT and AUTH [31]. SEND requests are used for content delivery, REPORT is optionally used for end-to-end acknowledgement of message delivery and AUTH is used in MSRP relay mode authentication. An example SEND request message containing a single instant message:

```
MSRP j4l34uud7 SEND
To-Path: msrp://ocean.com:12763/p33deirfwy2;tcp
From-Path: msrp://sea.com:7654/geiuf4oi3yr;tcp
Message-ID: 85749983
Byte-Range: 1-19/19
Content-Type: text/plain
Hello, how are you?
-------j4l34uud7$
```

Any MSRP message always starts with characters MSRP, then transaction ID and method type. To- and From-Path fields contain the sender's and recipient's MSRP URI. The URI info is first exchanged with SDP session descriptors using SIP at session creation. It has two purposes, identifying IP address, with a fully qualified domain name (FQDN), and port for TCP connection setup and also identifying the MSRP session. The URI format is shown in Figure 3.7.

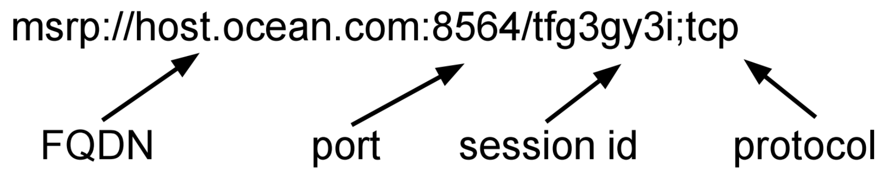

Figure 3.7: MSRP URI format [31].

Message-ID is an unique identifier for a single complete message. Byte-range identifies the range of bytes that are in this message and the amount of bytes of the complete message. Content-Type identifies the type of the message data, in this case being plain text and it is followed by the actual message. The end line contains a string of seven hyphens ('-'), transaction ID and either a '$', indicating the final message of this message-ID or a '+' character, meaning that this was not the final message for this message-ID.

If an image, video or file transfer is done, the message will be fragmented into multiple chunks. This transaction has a message-ID that is the same for all the chunks and the byte-range field indicates how many bytes are there in the complete message and how many in this message. Also the last character of the message will be '+' in all but the last message, which will have an '$' character [31]. This is the basic mechanism in MSRP for transferring different types of content with sizes varying from single text message to a file transfer that must be split into chunks.

4. BORDER GATEWAY

This chapter introduces the Border Gateway as a network element and the services it provides as a product. A more in depth look is taken to the aspects that are relevant for the RCS traffic encryption use case regarding the BGW, especially security related challenges and motivations.

4.1 Overview

The official name for the Border Gateway is Open BGW and it is a Nokia Networks product that is part of the Nokia IMS Border Control Solution [32]. Said solution is an integral part of the Nokia Voice over LTE end-to-end system. As seen in Figure 4.1, signalling and media plane handling responsibilities are divided to two different systems. The BGW handles the media plane and it is controlled by the CFX-5000 SIP session controller, that handles the control plane SIP messaging.

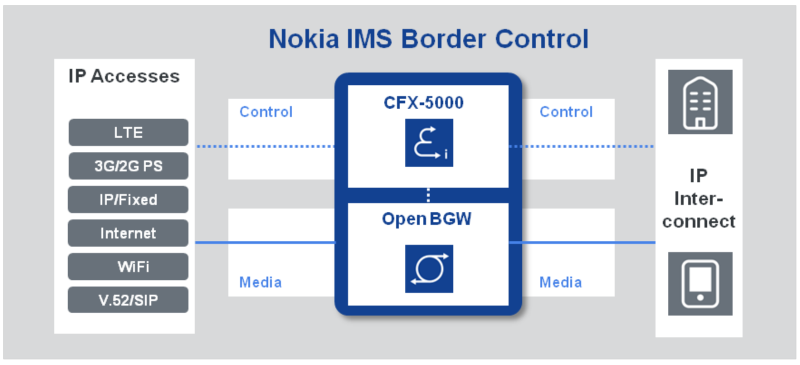

Figure 4.1: Nokia IMS border control solution [32].

Border Control, as the name suggests, is the entity between network borders. For example it can reside between the IMS core and an IP access network or between two peering IMS networks. It controls the delivery of all-IP IMS services which means that no IP content moves between the networks without control so it is responsible for network security. These aspects will be explained in more detail in Section 4.2.

The BGW, in 3GPP tems, implements IMS Access Gateway (IMS-AGW) [33], which is required when an access network is connected to the IMS. Other 3GPP functions are needed when, for example, two peering IMS networks are connected, but in that case there are no untrusted networks and additional encryption is not required and therefore it falls outside the scope of this thesis. Connected access networks can be untrusted depending on their nature. In Figure 4.1 there is a list of IP access networks and Internet is surely one of the possible untrusted access networks. The Border Control Solution is basically a Session Border Controller (SBC), which will be discussed in the next section.

4.2 Session Border Controller

Session Border Controller is an element regularly needed in common VoIP networks to exert control over the IP traffic. In many cases the need is to hide network topology and to protect the service provider or enterprise packet network from outside influence [34]. A general view of a VoIP network, which also applies to mobile networks, is seen in Figure 4.2. IP traffic needs to get to the core network, but it must do so in a tightly controlled way. The need for control is even more important when the SBC is handling the border of a public access network, like the Internet. In that case it must deal with a number of security considerations:

Denial-of-Service (message flooding)
Intentional malicious traffic, whose goal is to prevent others from using the service, is a common problem that is hard to solve. Unintentional flooding can also be a problem, with almost the same impact as intentional flooding. For example, after a large-scale power outage a flood of registration requests is automatically sent when the devices gain power.

Malformed messages
Intentional or not, incorrect protocol messages can cause problems if the networking stack cannot handle unexpected inputs. It can cause the receiving endpoint to behave in a unexpected manner; usually a DoS-type situation.

Traversing firewall or NAT
NAT devices are prevalent in network borders to hide network topology, which the SBC must be able to handle when controlling connections to different endpoints.

Regulatory mandate (lawful interception)

Governmental regulations often require the option to lawfully intercept calls and the SBC must provide the means to do that, at least in carrier provided services.

Ensuring QoS

Ability to prioritize traffic and ensuring service quality is a generic issue that cannot be taken for granted in IP traffic.

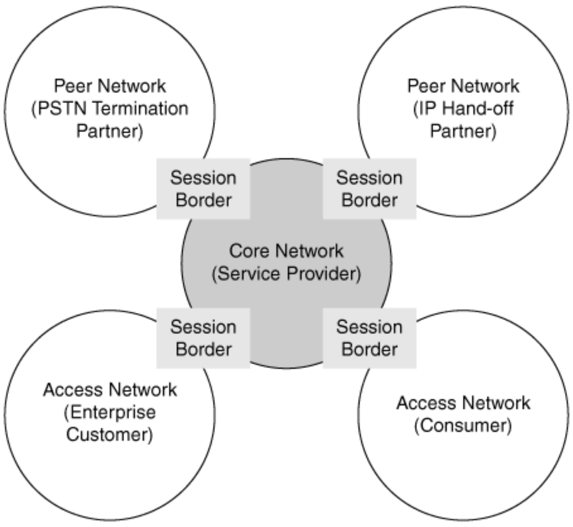
Peer Network (PSTN Termination Partner)
Peer Network (IP Hand-off Partner)
Session Border
Session Border
Core Network (Service Provider)
Session Border
Session Border
Access Network (Enterprise Customer)
Access Network (Consumer)

Figure 4.2: General view of SBCs in a VoIP network [34].

When a UE device initiates a session, e.g. using a RCS service, it must send a SIP message to start the process. This initial step is the most common situation where the need for an entity like the SBC is needed. Anyone connected to the Internet can send a SIP message to the SBC, as it must be open for accepting SIP messages. As the Nokia Border Control Solution is a decomposed SBC, where the signalling and media transport responsibilities are divided to different devices, the only relevant security considerations regarding this thesis are the possible threats on the media plane traffic, specifically the NRT traffic. This traffic consists of different protocols that are terminated on the BGW and it must be able to handle anything that might arrive from the UE. Possible malformed messages are the only threat that must be mitigated regarding the BGW and NRT traffic and this is explained in more detail in Section 5.1.

A more detailed view of the usage and position of the BGW at the border of an LTE packet core and a public access network with other Nokia network elements is seen in Figure 4.3. The mentioned NAT devices are not visible here because the BGW, as a carrier grade element, comprises in its entirety of many common networking devices, including NAT.

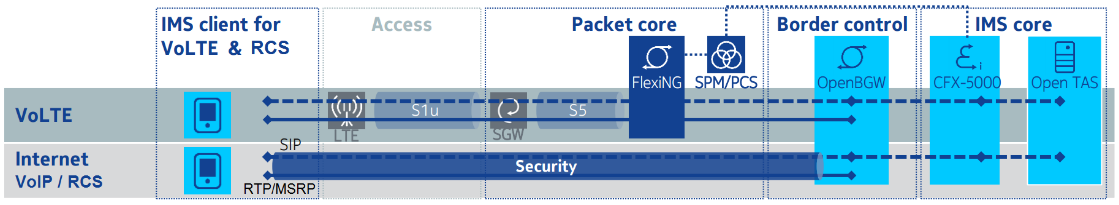

Figure 4.3: The BGW as an IMS Access Gateway handling trusted and untrusted networks, adopted from [35].

Figure 4.3 shows the operation of the BGW with two different access networks. Operator managed LTE packet core network doesn't need additional media plane encryption, which is seen in the upper part of the figure. This thesis work studies the non-real-time MSRP traffic that flows between the UE and the BGW and is encrypted due to the untrusted access network. The encrypted connection over the untrusted network is terminated on the BGW and the traffic is forwarded to the IMS core, which can be seen in the lower part of the figure.

4.3 ATCA platform

The whole BGW ATCA platform houses different functional units, which are operating on their corresponding blades on the ATCA shelf. The only relevant blade for this work is the DSP blade, which houses hundreds of individual DSP cores. A single ATCA based BGW can be allocated to serve hundreds of thousands of subscribers, so multiple blades of DSPs is required to handle the traffic that carriers face. The BGW application software that handles the encryption and packet forwarding runs on a single DSP core and it serves only a handful of concurrent connections.

The performance of a single DSP core is under analysis in this thesis because it does the media plane operations of the BGW and is only multiplied to increase capacity. So once the capabilities of a single DSP is known, then the overall performance of the whole system is also known. The system is getting simpler when only looking at a single core because then the architecture resembles that of any other traditional embedded system according to the model in Figure 4.4. In the simplified architecture there is a single DSP is doing the computing in the hardware layer, a real-time operating system (RTOS) in the system software layer and the BGW application forming the application software layer.

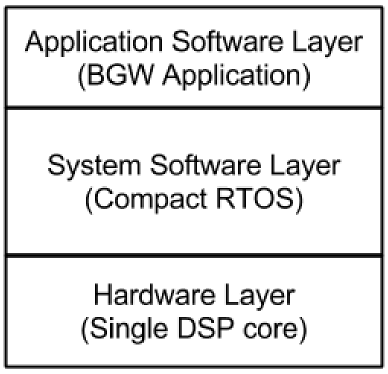

Figure 4.4: Embedded Systems Model, adopted from [36].

The actual model of the DSP device that was used in this thesis is Texas Instruments TMS320TCI6486 [27]. It is designed for telephony infrastructure applications and houses six physical cores operating at 625MHz. One of these cores will do the computing when the system performance is measured. The clock rate of the CPU in itself doesn't necessarily reveal much about the capabilities of the system because CPU instruction sets, system memory bandwidth and the type of the calculations are also a defining factor. This DSP in question has specific functions which it was designed to fulfil with great efficiency, but e.g. AES encryption calculations doesn't happen to be one of them.

4.4 Virtual platform

The virtual platform should function identically, as the embedded version, but it must do so by running on a virtual machine in the cloud. When comparing the virtual system to the model in Figure 4.4, the differences are that on the hardware layer there is a x86 CPU and a Linux OS with additional layers on the system software layer. The application code is untouched, excluding some small platform related changes. It is important to notice that the most significant difference between the platforms is the CPU.

The same real-time requirements are also in effect here so the biggest challenge is to be able to provide a real-time environment for a Linux process. The challenge comes from the fact that Linux kernel related operations can and will cause interrupts to use the CPUs [37]. This is transparent to the running process but it will add latency which is not acceptable when the process is implemented on the assumption that nothing interrupts its operation. The general way to solve this is to allocate one core, from a multi-core system, that the kernel interrupts never use and also make sure that no other process is running on that core. There are a number of ways to achieve this but doing so in a virtual machine does not guarantee that the actual physical core, which the virtual CPU uses (vCPU), is not used by other virtual machines or even the host platform. This is a general issue with cloud technology which Telco cloud needs to be able to do, but falls outside the scope of this thesis.

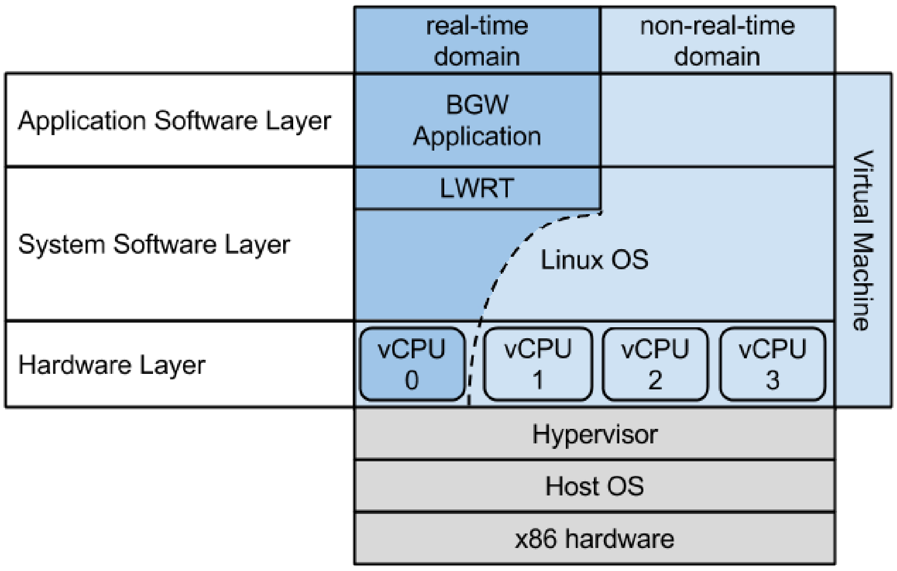

Figure 4.5: Virtual platform architecture overview.

In this work, the full control of a single CPU core was achieved using a Light Weight Run Time (LWRT) execution environment [38]. It is a commercial off-the-self product that partitions the system to a real-time and non-realtime domains and thus makes it possible to allocate cores to either domain. A high level architecture view of the cloud system is in Figure 4.5 with depictions of the same layers as in Figure 4.4.

The used CPU was Intel Xeon E5-2665 with 2.4 GHz core speed [39]. The clock rate is roughly four times faster than the rate on the DSP platform. As stated on Subsection 3.2.3, all modern Intel processors have an instruction called AES-NI that provides optimized instructions for AES calculations and its impact on performance should be significant.

5. TESTING METHODS AND SOFTWARE

This chapter introduces the aspects that will be tested and the used software. The testing methods are explained with detail so that they can be reproduced. These methods can be applied to any similar network element so very system specific details are left out.

5.1 Security

The Border Gateway is located at the border of the core and the interconnected IP access network. Such access network can be the public Internet, but the BGW is not directly seen from the access network side as network topology is obfuscated with the use of NAT devices. Any feature that is implemented on a network element such as the BGW must be done in a way that secure operation isn't compromised, although nothing is truly secure but every reasonable action should be taken to try and ensure it.

Any data that is by its nature considered untrusted and processed or manipulated in any way by the system has the possibility of causing unwanted operation. In this case, the untrusted data that effects the operation of the BGW are the network protocol message headers that originate from the UE device and are terminated on the BGW. These protocol headers are read and the system acts accordingly. If the protocol messages are malformed, by accident or deliberately, and the protocol implementations aren't robust, bad things can happen.

The BGW application software can run either on the embedded DSP or the cloud platform. Even though the software may be mostly similar, the two different platforms use different implementations of the network protocols. Varying kinds of network protocols are in use in the normal operation of the BGW but on the scope of this thesis work we are interested on the protocols that are used in the RCS messaging use case and also terminating on the BGW itself. Protocols that we are interested are from bottom to top: IP, TCP, TLS and MSRP.

The BGW application is running over a Linux platform in the cloud. TCP and IP protocols are implemented in the Linux kernel and these can be considered secure and pointless to test for any vulnerabilities. Linux has been around for a long time and used in untold secure servers so it is outside of the thesis scope to search for any vulnerabilities concerning Linux.

On the DSP side things are different. The platform the BGW application is running on the DSP is not considered to be a public commercial product. This means that we cannot straight away presume that it has already been rigorously tested. As TCP and IP are fundamental network protocols they are, also in this case, implemented in the kernel of the underlying operating system.

The BGW can be presumed to be behind a standard NAT router meaning that IP packets arriving to the BGW originates from the NAT device and therefore can be considered non-malicious in nature. Regardless, IP is the fundamental underlying protocol and can be tested with little extra effort. TCP packets on the other hand are originating from the UE device and are terminated on the BGW, so the TCP implementation must be robust and be able to handle any scenario without problems.

TLS protocol is also terminated on the BGW and the protocol messages are also generated in the UE end. The same level of robustness is expected from the TLS implementation. In both platforms the TLS implementation comes from the OpenSSL framework as it is the defacto SSL/TLS implementation and is used globally on most Internet related services. We are not interested to test OpenSSL in itself, as it also is outside the scope of this work, but as TLS is the top level terminating protocol, the robustness of the system and correct error handling is efficiently tested through TLS protocol testing.

MSRP is an application level protocol and from the BGW point of view is considered just a payload inside the TLS frame so anything related to MSRP protocol is hard to see affecting the operation of the BGW in any way. Malformed or intentionally malicious MSRP messages should not cause problems for the BGW. As the MSRP messages can be considered personal data between the recipients, there is no lawful reason to even look at the actual MSRP payload. Taking everything in to consideration, IP, TCP and TLS implementations are the ones that will be tested.

The robustness of the network protocol implementations are only a concern on the DSP platform as the used proprietary software is specifically made for the embedded platform and not openly available to the public. In the cloud this is not the case as all the software can be considered public and therefore robust enough.

The source codes of the tested protocol implementations are not available so we must regard the system as a black box. We can insert inputs and receive outputs from the system but cannot see what happens or why. The goal is to find an input that causes the system to behave in an unexpected manner. As we don't have any idea what to search for, the only option is to try all the possible variations of the inputs. The more complex the protocol, the more different variations there are.

Creating thousands of test cases manually where the sent message differs from the correct protocol format is not practical as it requires a lot of effort and still we cannot be sure if all the right combinations were tested. This kind of testing is called fuzzing and there are dozens of commercial and free fuzz testing frameworks available. Fuzzing is a negative software testing method that feeds a system with malformed and unexpected input data in order to find unknown vulnerabilities.

The testing was done with Codenomicon Defensics software and it can be considered the most comprehensive fuzz testing framework available. The OpenSSL Heartbleed bug was discovered trough Codenomicon testing [40]. With Codenomicon software it is quick and relatively effortless to find defects that can be triggered by malformed inputs. There are testing suites available for almost all the common network protocols. These suites contain possibly millions of machine generated test cases of the protocol in question.

No matter what protocol is being tested, Codenomicon tests usually have the same basic idea as illustrated in Figure 5.1. In a simple scenario the test cases can be sent without any preparation of the tested system. In our case the system requires control plane signalling to prepare the connection setups so our test system too will need to send signals to the system so that it will be ready to accept incoming connections.

After the system is ready for an incoming connection, malformed packets, incorrect protocol messages, repeats of correct or incorrect messages or anything in between are sent to the system. After every case a valid message is sent as an instrumentation to check, if the test case had any ill effects on the system. If the system responds correctly and on time, the test passes. If the system is unresponsive or takes longer than specified to answer, the case can be deemed a failure.

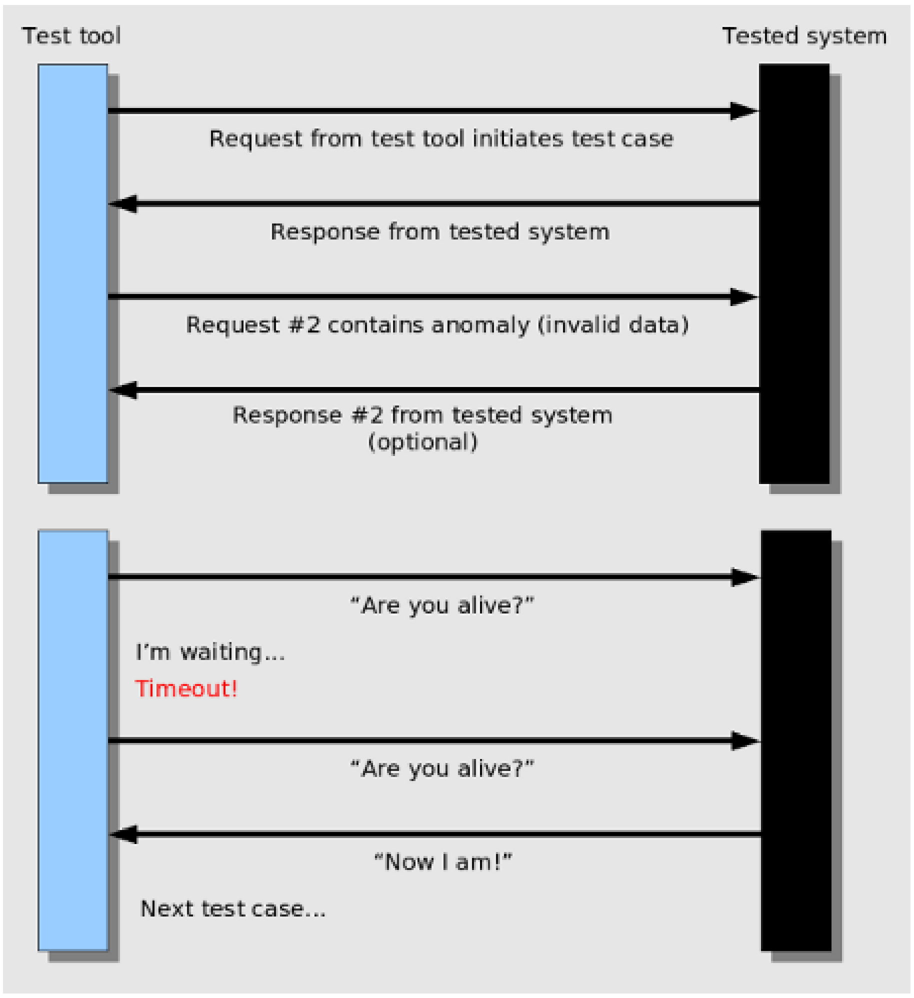
Test tool
Tested system
Request from test tool initiates test case
Response from tested system
Request #2 contains anomaly (invalid data)
Response #2 from tested system (optional)
"Are you alive?"
I'm waiting...
Timeout!
"Are you alive?"
"Now I am!"
Next test case...

Figure 5.1: Codenomicon testing method.

5.2 Performance

Testing the system's performance is done for multiple reasons. Firstly, to make sure that the system's performance is at an acceptable level, so that it can serve the amount of connections that it is supposed to be able to serve. Secondly, how do the two different platforms compare? We start to see the effects of available computing power and how it scales on both platforms.

Network throughput is one of the most important variables when starting to measure the overall performance. Data that is being forwarded by the BGW is either being encrypted or decrypted, depending on the direction. So the limiting factor on how much data can traverse through the system is most probably the systems ability to encrypt or decrypt data. There are no specific performance goals but the only variable that is defined is the amount of simultaneous connections. The specifications were decided with the DSP platform in mind as the virtualized implementation can still be considered more of an prototype than a ready, releasable product. Nevertheless, the software is as identical on both platforms as possible.

The amount of latency that the BGW induces on the messages that traverse trough it, is also a measurable variable. Although, the MSRP messages are non-real-time traffic and latency is not an issue, as long as it is in acceptable levels. The QoS model for IP-based traffic is normally a best-effort model, so it is important to verify that all the connections receive an equal amount of resources, if the BGW ends up being the limiting factor. Measuring the individual message latencies and their spread in different scenarios is one way to make sure that all the connections are treated equally.

Lastly, measuring the used processor resources that are utilized at any time is vital, as it is probably the limiting factor for total throughput. TLS handshake is performed for every new connection and it is an expensive operation, but its effects will be minor overall, assuming they are not performed constantly. This assumption may be dangerous regarding system performance, but there simply isn't any information available on the probable use profiles of the RCS service as it is still in the development phase. The assumption that the results are relying on is that the system needs to perform handshakes so sporadically, that they will not affect the performance and can therefore be ignored as a factor in the measurements. So the only cryptographic operation is the symmetric encryption/decryption that is performed for the passing data.

Planning comparable measurement methods for two different kind of environments requires that the used tools can operate on both platforms. Also, the environments are not fully controllable, especially the systems around the DSP platform. We basically must make do with what tools are already available as installing additional software is probably not worth the effort. The common factor for the test environments on both platforms is that there is a Linux system with network access to the tested system. Even though they are different Linux distributions, there are tools that are included in most of them by default.

We need to simulate the UE devices, so we need two endpoints that are able to send and receive TCP traffic, encrypted or not. We also need to be able to generate MSRP traffic in sufficient amounts so that we can find the maximum throughputs of the systems. This all can be done with the following simple tools that are available in most Linux systems:

openssl s_server

A generic SSL/TLS server to accept incoming connections.

netcat

Versatile and feature-rich networking utility for reading and writing to TCP/IP connections.

dd

A Linux command primarily used for converting and copying a file.

I/O redirection

It is relatively straightforward in Linux to direct the output of one program as a input to another program, with bash commands like "|" or ">".

The scenario where the measurements will be taken is one where a UE device from a trusted network sends as many MSRP messages as the BGW will handle. The BGW will encrypt the data and send it to the UE that is located on the, supposedly, untrusted side. Generating as many such connections as possible should impose the BGW as a bottleneck and reveal its performance characteristics.

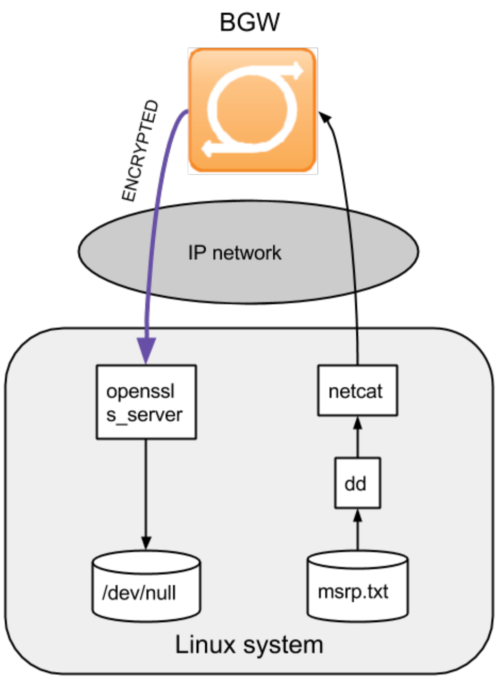

Figure 5.2: Test setup configuration.

The measurements procedure goes as follows:

1. Required amount of OpenSSL s_servers are launched and they begin to listen to their port for an incoming connection.

   ```
   for x in range {50000..50063}; \
   do (openssl s_server -cert mycert.pem -accept $x > /dev/null) \
   & done
   ```

   This simple Linux bash script can be run from the command line and starts 64 instances of s_server to listen ports 50000 through 50063 and directs the output from them to /dev/null file descriptor, so the large amounts of incoming data will be discarded straight away as there is no need to store it. A certificate file is also needed by the s_server.

2. With control plane signalling messages, the connections are initiated on the BGW so that every s_server instance receives an incoming encrypted connection from the BGW so there will be 64 TLS connections between the BGW and the s_server instances. After this the BGW will be listening on 64 different ports for the connections that will be paired for the already established encrypted connections. For this test, the consecutive ports are used for easier management.

3. Now we need 64 instances that are able to send continuous non-encrypted MSRP traffic.

   ```
   for x in range {10000..10063}; \
   do ((while true; do dd if=msrp.txt; done) | netcat 12.248.204.155 $x) \
   & done
   ```

   This creates 64 netcat instances that connect to the given IP and to ports 10000-10063. With every instance there is a dd command in an infinite loop copying the file mrsp.txt, that contains lots of short MSRP messages, as an input to netcat which sends it forward in chunks as TCP packets.

4. After this the MSRP messages are sent indefinitely to the BGW that encrypts it, forwards it to the corresponding connection that upon receiving it discards it. To get measurements with different amount of connections, the connections are torn down, one by one, through control plane signalling. This way the effects of handshakes are not visible as they are performed at the start of the connection in step 2. The complete setup is illustrated in Figure 5.2.

Depending on the achieved throughput, there is a possibility that dd cannot read the file from disk fast enough and that becomes the defining bottleneck. As the data is sent over a network interface, that too can possibly be a restricting factor. If the BGW achieves maximum CPU utilization, then all the other possible bottlenecks can be ruled out. The 64 connection pairs are running on a single computing core and at least on the DSP side, a single DSP core is not expected to achieve high symmetric encryption throughput, as found in Section 3.2.3.

Latency can be measured by sending a timestamp inside a MSRP message payload and substracting it from the current time when it is received by the other end. This was done by getting a timestamp with millisecond precision from the `$(date)` bash variable and with the previous steps and simple scripts that did the subtracting. This is an unreliable method as operating system interrupts and process handling can have an effect on how close the results are to the actual latency. The accurate way would be to monitor the timestamp that is generated by the network adapter hardware upon sending and receiving packets. We are after for possible high latencies, that would be visible in text message communication, so nanosecond accuracy for timestamps is not needed.

Latency will first be measured by initiating 32 connection pairs, where the trusted side sends one MSRP message, with the timestamp as payload, randomly between 1 and 10 seconds, in an attempt to mimic an actual chat session. Data is collected for about 10 minutes to get the general trend visible for the latency graph. After this, additional 32 connections pairs are added that generate as much load as possible to stress the BGW. The goal is to see if high load connections increase the latency of other connections. Comparing results from the DSP and the virtualized platform could have given some insight on how the different platforms behave and if there is any noticeable difference regarding latency. Unfortunately, latency measurement was only possible in the virtual platform, as this method of latency measurement proved too unreliable in the DSP test environment. Timestamps from the `$(date)` bash variable were too inaccurate in that environment to be of any use in this scenario.

Getting the CPU and throughput values for every connection amount was done manually. CPU utilization info is queried from the BGW with control plane signalling mechanism and the throughput is the network adapter transmit rate of the system that is generating the traffic. It was noticed during testing that encrypted data traffic was noticeably larger, due the overhead that TLS protocol adds. So all the throughput values were collected from the trusted side that is sending non-encrypted messages. 64 data points can be collected by hand, with hundreds of connections, an automated script would have been needed, but for this amount it was simpler to get the data by hand instead of making a script for it.

6. RESULTS AND DISCUSSION

The results are presented in this chapter and their significance is discussed. First the outcome of security related testing is gone through and after that latency and throughput benchmarks are discussed and compared between the platforms.

6.1 Security

Before the security related testing results are discussed, it is important to mention that a certain level of abstraction is maintained here because of the nature of the tested object. The Border Gateway is a commercial product that is responsible, for its own part, for maintaining a secure network border, so going in to too much detail on the found vulnerabilities is avoided. Security related testing was originally planned to provide results and to be compared between the platforms to find the possible differences and challenges when transitioning to a cloud platform.

As reasoned in Section 5.1, only the DSP platform needed to be tested so this became purely an exercise on finding vulnerabilities on the DSP platform. The parts that was tested were IP, TCP and TLS protocol implementations of the DSP platform. Before testing was started there was no notion of anything being wrong with the tested software. There were no known problems or bugs with the protocol implementations so the goal was basically to try and find a way to get the DSP platform to do something that was not designed by only having a direct network access to it.

If it would be possible to affect the correct operation of the system just by having a network access to it, the system could not be considered very secure. Using malformed IP packets to exploit vulnerabilities of a device that is behind a NAT should not be possible because the NAT usually replaces most, if not all, the IP header fields with new ones, but this is not the case for TCP. In theory, an attacker could initiate a normal chat session with somebody and modify some of the outgoing TCP packets carrying the MSRP data to have malformed fields in an attempt to exploit a vulnerability in the TCP/IP stack of the terminating endpoint, in this case the BGW. As stated in Section 3.1, the attacker can be presumed to have unlimited resources and time, so they will find a way, if there is one to be found, theoretical or not.

The Codenomicon testing software comprises of test suites for many different protocols. There are many configuration options that need to be correct so that the tested platform can communicate with the test suite. Depending on the tested protocol, the complexity of the required configurations vary. Testing was started at the bottom of the protocol stack to ensure that we have a complete picture of the viability of the underlying protocol implementations once we start to test the upper level protocols.

Testing any protocol requires that the tested system has a service that is willing to communicate with the protocol in question. This is easy to achieve, at least for the IP test suite. Any system that has an IP address also implements Internet Control Message Protocol (ICMP) as it is one of the main protocols of the Internet Protocol Suite. The ICMP echo request, also known as ping, must be answered with an ICMP echo reply and this mechanism is what the Codenomicon IP suite was selected to use to test the system's TCP/IP stack. Basically, the Codenomicon IP suite sent malformed ICMP echo requests to test the IP stack of the tested system.

There were around 170 000 test cases that were performed by the software and 17 of them were found to get the IP stack of the DSP in such a state, that it became unresponsive. These cases required that there were specific errors on two different fields of the IP header, but still all of the cases were exploiting the same flaw in the code. This type of vulnerability could be used against the system as a type of denial-of-service (DoS) attack as all IP-based traffic would cease once the IP stack stops to operate after it receives the correctly malformed IP packet.

A service that accepts incoming TCP packets and acts on them is also required for the TCP test suite so there needs to be a port that is actively listened by a process for incoming TCP connections. The DSP embedded platform could be accessed through a Telnet interface and this meant that there is a process listening for a specific port for Telnet connections. Performing a TCP handshake to this port is possible even though we have no intention on using the Telnet service itself, which makes it possible to test the TCP implementation of the DSP. The port and the type of service we are supposedly using is configured to the Codenomicon TCP test suite so that the suite can send a valid Telnet packet sequence as a valid test case to see if the DSP is still responding after a normal test case.

Around 340 000 test cases were ran and the failed ones could be divided to cases that fail on their own and cases that require other cases to be sent prior to them to get the system to become unresponsive and the tests to fail. The first group of cases were very similar to the ones found in IP testing as incorrect information on a specific TCP header field made the TCP/IP stack become unresponsive. Identifying the problem is straightforward once a single test case is identified that fails every time, but the situation is not so clear when test cases require other cases to be sent before them to get a result. The decisive hint proved to be the fact that every case that required prior tests required the same number prior tests. Certain cases made the Telnet service to not free up the allocated memory for the connection in question and eventually the memory ran out and undesirable things happened.

The goal was to test the TCP implementation through the Telnet service, not the service itself. Getting failing protocol tests cases due to application bugs made it harder to analyse the results and took extra time when answers were searched in the wrong places. The Telnet service was used for its convenience as it was the only readily available service listening for TCP connections in the embedded platform. The positive outcome from this was that a system breaking bug was found, even though the Telnet service is only used for debugging purposes and is normally disabled.

The findings from the TCP/IP stack testing were reported to the developer and they made the required corrections and supplied a new version. The original tests were executed a second time to verify that system can handle all the thousands of variations of malformed protocol messages. After this the TCP/IP stack is considered secure enough for our purposes.

The last protocol to be tested was TLS and the only service that had TLS capability was the actual BGW application software. Every test case closes the connection at the end, so between every case, a new connection request needs to be sent through signalling messages so that the BGW actually accepts the connection made by Codenomicon test suite. This requirement made the test suite configuration more complex, but once everything was set up correctly, the tens of thousands of cases would execute slowly but automatically.

The TLS implementation comes from OpenSSL version that was modified to be used in the DSP embedded platform. Regardless of the modifications, the actual cryptographic functions were untouched and nothing was expected to be discovered through Codenomicon fuzz testing. This assumption proved to be right, as only minor error handling corrections were made to the application software based on the TLS testing results. Not a single test case failed due to OpenSSL.

Codenomicon Defensics fuzz testing platform proved to be an invaluable tool for testing protocol implementations, but the biggest limiting factor that hindered testing was time. Depending on the tested protocol, a single test suite instance could execute about one to two test cases per second. The full test profiles of the suites can have millions of test cases and executing them all is not feasible. For example, full test profile of the IP test suite contains 3,2 million tests cases and it could take weeks for all the tests to execute. This can be overcome by running multiple test instances simultaneously, but a single instance required up to 1GB of system memory so only a couple could be used at the same time as the system ran out of memory. It was stated in Section 5.1 that every reasonable action should be taken to ensure security and running week long test runs is bordering unreasonable. The default test profiles provided a good coverage while being manageable as they needed to be run multiple times during testing.

6.2 Latency

Latency measurements were only performed for the virtualized platform. One virtual machine was running the application software and other separate instance was sending and receiving the data, basically the same setup as in Figure 5.2. A baseline measurement was done to measure the delay that is present in the cloud environment. The delay between the two instances and the probable inaccuracies that are caused by the measurement method, added up to be 1 to 2 ms. So this amount of delay will be present at minimum and anything above it will be induced by the application.

First the 32 connection pairs were set up that only sent a short text message randomly between 1 and 10 seconds, so system load was very low. These latency values are plotted in Figure 6.1. No decimals were calculated which causes the visible vertical lines in the distribution. With 5000 individual messages the latencies seem to distribute very evenly. Roughly all the messages arrive after 2 to 12 ms. If we subtract the known error of 2 ms, then the latency would be 0 to 10 ms. This is a good result, as it is known that the software forwards the data every 10 ms. So if a message arrives just before the 10 ms interval, then it is forwarded almost immediately and the latency is what is caused by other factors, in this case being the 2 ms. And if it arrives just after the interval, then the packet waits for the full 10 ms before it is forwarded. This behaviour is clear in the graph.

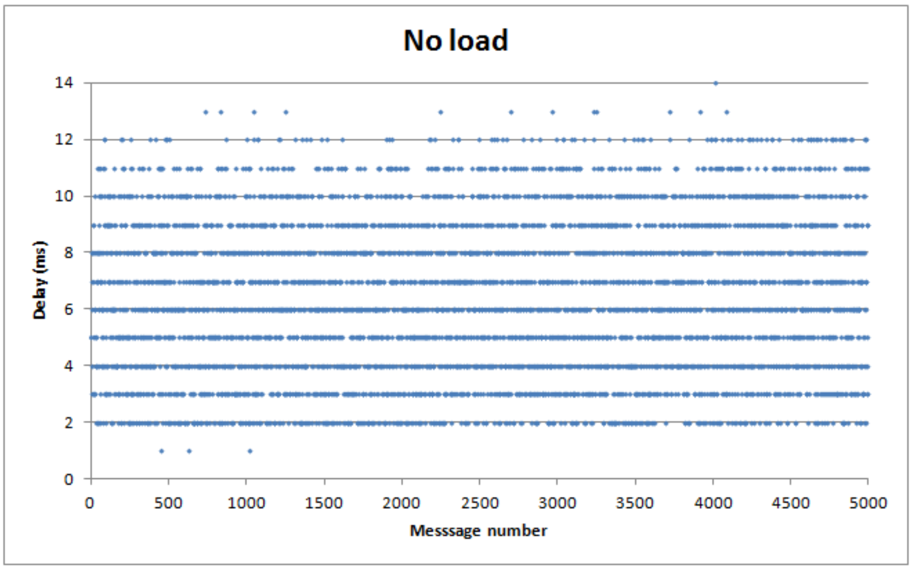

Figure 6.1: Message latency measured with no additional system load.

The result we are after is what happens with the latencies when the system is under heavy load. This is the same configuration as previously with the difference that there are additional 32 connection pairs sending data with no artificial limits. System CPU load was around 70-80%, so close to the maximum utilization. The result is in Figure 6.2 and it is quite surprising. The latencies actually drop under heavy load. This is caused by the mechanism in the software that lowers the interval from 10 ms to a minimum of 1ms, depending on the load level.

With couple of exceptions, all message latencies seem to distribute between 3 to 6 ms, so not even close to anything that would be visible to the users in chat sessions. The behaviour we were after is not present so these latency measurements proved to be just a verification that every connection is treated equally. Another partial failure was that the measurement technique didn't work in the environment used in the DSP application testing, as baseline measurements showed that the results were too unreliable. Because there is no hint of increased latencies, overhauling the measurement technique just for the DSP environment would not be worth the effort.

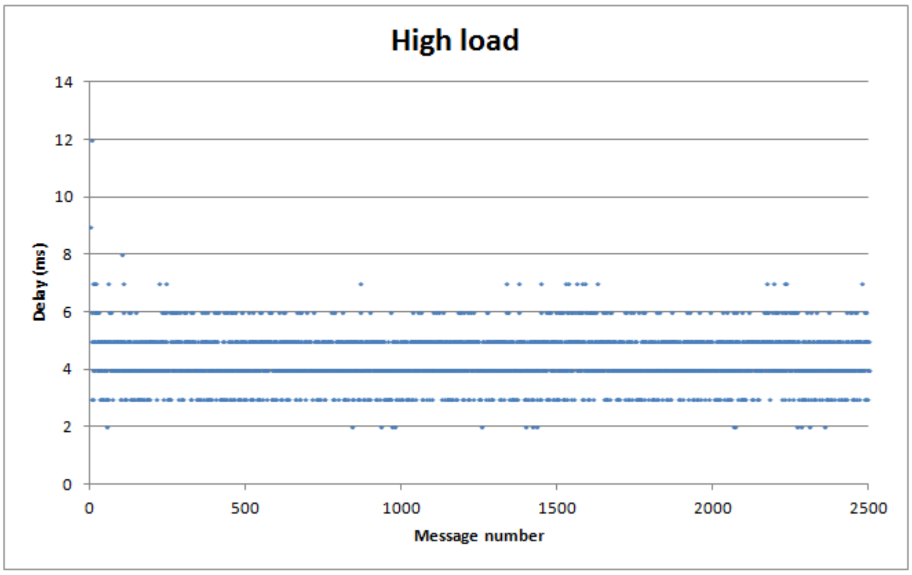

Figure 6.2: Message latency measured with high system load.

6.3 Throughput

The performance characteristics of the platforms are important information regarding the product's capabilities and how many users can be served per one computing core, but regarding this thesis work, we are more interested on how they compare and what causes the differences. To get comparable results from both platforms, the measurement process must be as similar as possible. The used environments are generally the same, meaning that the used tools and Linux distributions are the same but most likely different versions. Regardless of these differences, the results should be accurate enough so that they can be compared.

After the connections are set up, all that the BGW is doing is basically just encrypting data with AES cipher with a key size of 128 bits, because that symmetric algorithm was used by default. No decryption is performed because of the one way traffic flow of the test, but there should not be any noticeable difference if data is decrypted or encrypted [41]. If the data is not encrypted, then all the computational costs come from the cost of forwarding packets from one port to another. In such a case, it would mean that both UEs are located on a trusted network and in such a case, the traffic is just relayed over the BGW. Measuring performance without encryption is not an actual use case regarding this work, but it gives additional data about the cryptographic capabilities of the system.

Before going through the results, we know from Subsection 3.2.3, that the platforms should be capable of processing, when converted to bits, about 4,5 Gbit/s in the cloud and 0,3 Gbit/s in the DSP platform. This is a synthetic benchmark and does not have much significance when measuring the real performance of the system, but what we can deduct from this, is that at least for the modern x86 processor, the rates are so high that many other factors will be limiting the performance long before the theoretical encryption rate is achieved. Also, when fewer CPU cycles are consumed by the encryption process, more cycles are available to be used somewhere else and the performance scales with the traffic accordingly.

The measurements were done with and without encryption. In the latter case, the only difference in Figure 5.2 is that both sides have a netcat instance as there is no encrypted traffic. Every graph have three plots, CPU utilization, total throughput and single connection throughput. Every plot consists of 64 data points that were collected manually for every number of connections. CPU and throughput values do not stay static so all the data points are an approximation of the current average value collected by hand so there is naturally some inaccuracies involved, but with multiple data points the overall trend should be accurate enough. The single connection plot is derived from the total throughput graph by dividing the it with the number of connections, but this is a valid result as additional testing was performed to validate that every connection receives an equal share of the total throughput.

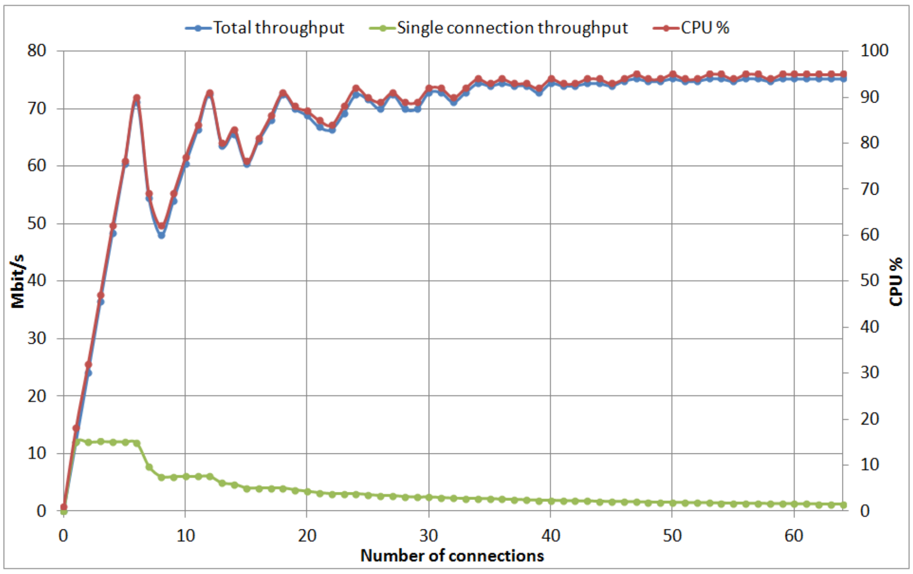

Figure 6.3: DSP system performance with no encryption.

DSP platform graph without encryption is in Figure 6.3, which is a logical place to start inspecting the results. It becomes obvious that CPU and throughput plots are nearly identical and any big differences most likely occur from the approximation of the collected values. Some unknown and undefined behaviour takes place at the beginning of the graph. There is no obvious reason why the throughput should drop from 70 Mbit/s with 6 connections to 50 Mbit/s with 8. After 30 connections this behaviour seem to disappear. One important thing to notice is that the maximum rate for one connection is capped by the application at 12 Mbit/s so that is the reason why one connection at the beginning doesn't get more than that.

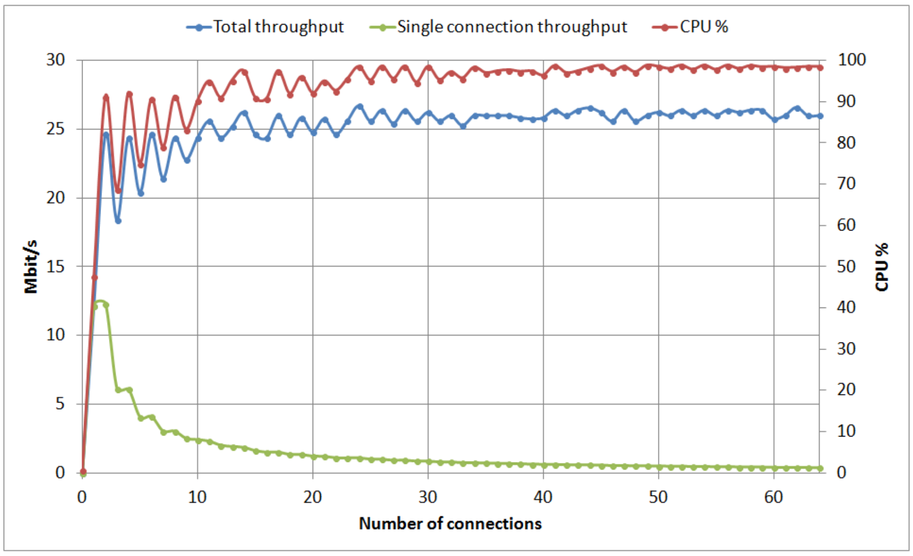

Figure 6.4: DSP system performance with encryption.

More relevant data is in Figure 6.4 where encryption is enabled. Maximum total rate without it seemed to be 75 Mbit/s and with encryption it is around 26 Mbit/s. A three fold decrease in performance when data is encrypted brings the total rate so low that only two connections can be served at their maximum rate. At 64 connections the throughput of one connection is 0,4 Mbit/s and as such is very moderate. Every connection consists of two users who both can send files at the same time and if this happens, then the rate of one user is 0,2 Mbit/s, for both ingress and egress traffic. How often does a user send files instead of chat messages? It completely depends on the users how this service is received and utilized. There are 128 users for 64 connections and it only takes any two of them to send a file at the same time and the utilization of the DSP core is near maximum.

Results without encryption for the virtualized platform are in Figure 6.5. Everything is going as expected until around 40 connections are reached. The undefined behaviour we encountered earlier seem to be manifesting here also but on a larger scale. It seems that once the maximum CPU utilization is achieved and new connections cannot be served any more with the rate of previous connections, all rates are halved. Total throughput peaks at 410 Mbit/s with 37 connections so it is six times more capable than the DSP platform. As the virtual platform is operating with a core speed of 2400 MHz and the DSP with 625 MHz, there is a four fold difference in

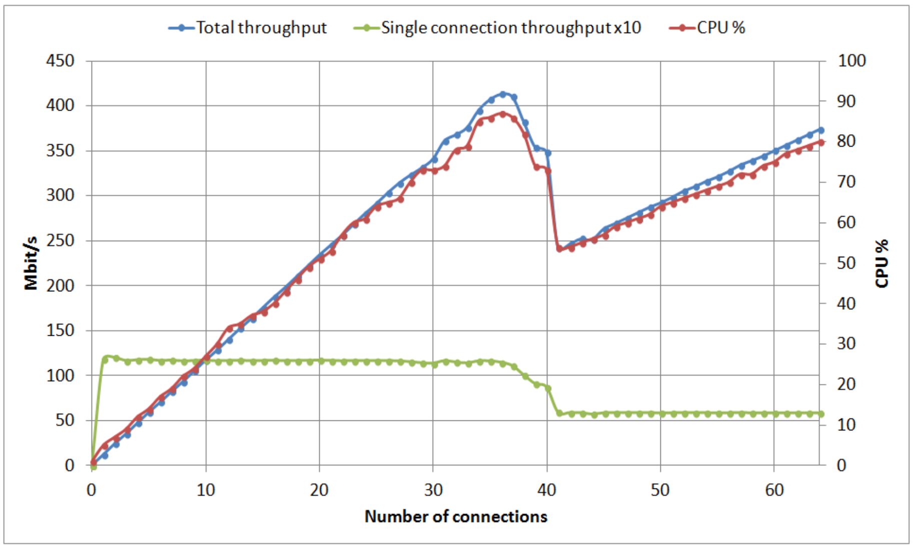

Figure 6.5: Virtualized system performance with no encryption.

MHz and therefore the performance difference doesn't correlate directly with MHz because of the differences in CPU architectures and other system factors. This is a good place to compare the effects of MHz as no encryption is used and the AES-NI instruction set does not create an advantage for the virtualized platform.

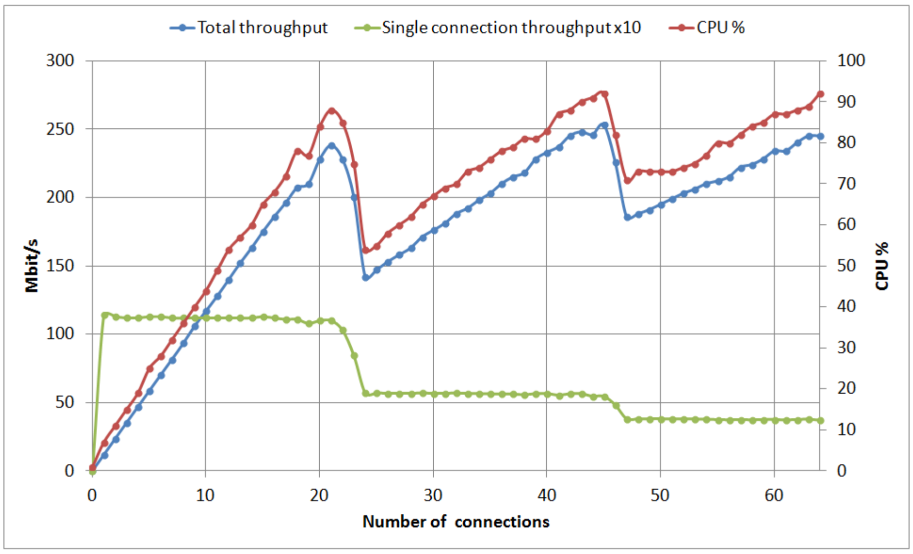

Figure 6.6: Virtualized system performance with encryption.

Final Figure 6.6 is the graph for virtualized platform with encryption. Throughput peaks at around 240 Mbit/s with 21 connections and again with 45 connections at 250 Mbit/s. The next peak is about to happen as CPU utilization approaches its limit, but the connection limit is achieved before it. In pure performance numbers, the application runs ten times faster on virtualized platform than it does on the DSP platform. Even if raw throughput ability might not be the deciding factor regarding the RCS use case, the virtualized platform can serve ten times more concurrent users than the DSP. The AES instruction set in the x86 architecture is the main reason why the performance difference grows when compared to the case without encryption, from six to ten fold difference. Four times more MHz and an instruction set that doubles the performance, so with crude arithmetic the result would be eight times more performance, not quite the actual result but still relatively close.

The mentioned undefined behaviour is tied to the moment when CPU utilization cannot grow any more. This happens ten times faster on the DSP and hence the plots differ. If the test would have been performed on the virtualized platform with 640 connections, then the graph trends would probably look alike. The testing method would scale to any number of connections without extra effort, but gathering 640 data points by hand would not be practical. This also works in the opposite direction, if we compare the DSP performance with 6 connections to the virtualized platform with 64, then the trend becomes visible. This graph is in Figure A.1 and it is very similar with Figure 6.6, which verifies that the both platforms perform similarly when the performance difference is taken into account. This also applies to the case without encryption when the original DSP graph is scaled down to one sixth of the connections to 11 connection in Figure A.2, as it is very similar as Figure 6.5.

Overall, the performance differences between the platforms proved to be quite broad. Directly comparing the performance of a single computing core between the platforms might not be quite fair, as they are very different by nature. A single ATCA-based BGW houses hundreds of DSP cores, and as a carrier grade hardware, costs tens of thousands of euros. How much does the same amount of capacity cost when it is taken from the pool of resources in the cloud? How much electricity both platforms consume? These questions fall outside the scope of this thesis, but the results should provide information when decisions are made regarding the viability of Telco cloud and how capable the current generation of x86 hardware is for this type of telecommunication applications.

7. CONCLUSION

Using the differences in characteristics between the DSP's and virtualized platform's ability to handle the encrypted non-real-time traffic proved to be a suitable case to study the challenges of virtualization in telecommunication applications. The used processors are made for different purposes and the market forces behind the general purpose x86 processors ensure that the technology is advancing more rapidly, which means that the x86 processors are starting to handle the tasks that previously needed a special purpose processor, like the DSP, to make operation feasible. Handling the tight real-time constraints of telecommunication applications is one of the general challenges when moving to the cloud. The virtual machines use the pool of resources of the cloud and ensuring that the assigned resources are always ready to serve without delay to ensure carrier grade service requirements is one of the features that Telco cloud need to provide.

In this case the BGW application is performing the responsibilities of a Session Border Controller, which is basically just a terminating endpoint for encrypted connections and forwarding the packets to the non-encrypted side. These operations don't utilize the inherent advantages of the DSP over the general purpose processor, so they are almost equal MHz to MHz. The big differentiating factor is that the modern x86 processors are operating in a much higher frequency. In this case the processor in the virtualized platform had four times more MHz and also the AES-NI instruction set that halves the time the processor needs for AES operations, meaning that the symmetric encryption performance doubles. Overall, the application operating on a single CPU core was ten times more capable on the virtualized platform, meaning it can possibly serve ten times more concurrent users.

We are moving away from the closed world of proprietary software and hardware of the ATCA ecosystem to the cloud where most of the used software can be open source and thus used by a much larger audience. This minimizes the extra security efforts that are required in the case of closed software. The TCP/IP protocol stack is part of the operating system and in Linux it can be presumed to be robust. The stack is also part of the real-time operating system that the DSP platform is running on and through protocol fuzz testing flaws were unveiled that could have been used for DoS attacks, for example. Regarding security, it makes sense to use software that has been, and will be, the target of active attacks because if a flaw is discovered from them, the actual rewards for the attackers are high because the software is in use in many different systems, and therefore it must be securely implemented.

Testing was done with the latest x86 server processors which is possible because their update cycle is fast compared to digital signal processors. The DSP used in this thesis is just about to be replaced by a new generation, that has higher operating frequency and cryptographic instruction sets. So comparing processors that have many years between them might be unfair, but this is the nature of slow DSP update cycles. Though, studying the capability of the new generation DSP is reserved for possible future work.

REFERENCES

[1] PICMG. AdvancedTCA® Overview. https://www.picmg.org/openstandards/advancedtca/, 2014. [Online; accessed 2-April-2015].

[2] Nokia Siemens Networks. Open Core System - White Paper. http://nsn.com/system/files/document/open_core_system_technical_white_paper.pdf, 2011. [Online; accessed 11-September-2014].

[3] European Telecommunications Standards Institute. Network Functions Virtualisation - Introductory White Paper. http://portal.etsi.org/NFV/NFV_White_Paper.pdf, October 2012.

[4] NTT DOCOMO Technical Journal Editorial Office. Measures for Recovery from the Great East Japan Earthquake Using NTT DOCOMO R&D Technology. https://www.nttdocomo.co.jp/english/binary/pdf/corporate/technology/rd/technical_journal/bn/vol13_4/vol13_4_096en.pdf, March 2012. [Online; accessed 24-July-2014].

[5] GSMA. Dealing with Disasters: Technical Challenges for Mobile Operators. http://www.gsma.com/mobilefordevelopment/wp-content/uploads/2012/06/Dealing-with-Disasters_Final.pdf, June 2012. [Online; accessed 24-July-2014].

[6] ETSI. Network Functions Virtualization (NFV); Use Cases. http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf, October 2013.

[7] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS). http://www.3gpp.org/DynaReport/23228.htm, June 2014. [Online; accessed 4-July-2014].

[8] Rebecca Copeland. *Converging NGN Wireline and Mobile 3G Networks with IMS*. CRC Press, 2009.

[9] Miikka Poikselkä and Georg Mayer. *The IMS: IP Multimedia Concepts and Services*. John Wiley & Sons, third edition, 2009.

[10] Heavy Reading. Virtualization of IMS & VoLTE in Mobile Operator Core Networks. http://www.mavenir.com/files/doc_downloads/HR_Mavenir_VirtualizedIMS_0413.pdf, 2013. [Online; accessed 1-December-2014].

[11] Strategy Analytics (2013). *Handset data traffic (2001–2017)*. Strategy Analytics, June 2013.

[12] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, 2014. [Online; accessed 8-October-2014].

[13] Joydeep Acharya, Long Gao and Sudhanshu Gaur. *Heterogeneous Networks in LTE-Advanced*. John Wiley & Sons (UK), 2014.

[14] Nokia Siemens Networks. Wi-Fi integration with cellular networks enhances the customer experience. http://nsn.com/system/files/document/nokia_siemens_networks_wi-fi_integration_final.pdf, 2012. [Online; accessed 4-July-2014].

[15] Stoke. Wi-Fi Offload Evolution: A Practical Approach to Leveraging Unlicensed Spectrum to Offload Your RAN. http://www.stoke.com/GetFile.asp?f=59e137652e4fceb66dac895a1f9fe31b, 2012. [Online; accessed 13-October-2014].

[16] 3GPP. Architecture enhancements for non-3GPP accesses. http://www.3gpp.org/DynaReport/23402.htm, 2014. [Online; accessed 17-February-2015].

[17] Il Grigorik. *High Performance Browser Networking*. O'Reilly Media, September 2009.

[18] 3GPP. The Evolved Packet Core. http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core. [Online; accessed 16-December-2014].

[19] Miikka Poikselkä. *Voice over LTE: VoLTE*. John Wiley & Sons (UK), 2012.

[20] Rogier Noldus. *MS Application Developer's Handbook: Creating and Deploying Innovative IMS Applications*. Academic Press, 2011.

[21] D. Dolev and A.C. Yao. "On the security of public key protocols," proceedings of the ieee 22nd annual symposium on foundations of computer science, 1981. pp. 350-357.

[22] Rolf Oppliger. *SSL and TLS: Theory and Practice*. Artech House, 2009.

[23] TIM Trustworthy Internet Movement. Survey of the SSL Implementation of the Most Popular Web Sites. https://www.trustworthyinternet.org/ssl-pulse/, June 2014. [Online; accessed 3-July-2014].

[24] George Apostolopoulos, Vinod Peris and Debanjan Saha. Transport layer security: How much does it really cost? In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume Second, pages 717–725. IEEE, March 1999.

[25] Cristian Coarfa and Peter Druschel. Performance analysis of tls web servers. In *In Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*. Internet Society, 2002.

[26] Jeffrey Rott. Intel® Advanced Encryption Standard Instructions (AES-NI). https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/, 2012. [Online; accessed 21-July-2014].

[27] Texas Instruments. TMS320TCI6486 - Communications Infrastructure Digital Signal Processor. http://www.ti.com/product/tms320tci6486, 2014. [Online; accessed 28-November-2014].

[28] Texas Instruments. C667x Digital Signal Processors. http://www.ti.com/lsds/ti/dsp/keystone/c667x/products.page?paramCriteria=no, 2014. [Online; accessed 11-September-2014].

[29] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.2., RFC 5246, IETF, August 2008. https://tools.ietf.org/html/rfc5246.

[30] Netcraft. Half a million widely trusted websites vulnerable to Heartbleed bug. http://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html, 2014. [Online; accessed 29-October-2014].

[31] Rogelio Martinez Perea. *Internet Multimedia Communications Using SIP: A Modern Approach Including Java Practice*. Elsevier Science and Technology Books, Inc., 2008.

[32] Nokia. Nokia IMS Border Control Solution. http://networks.nokia.com/portfolio/solutions/voice-over-lte/nokia-ims-border-control-solution, 2014. [Online; accessed 13-February-2015].

[33] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; IP Multimedia Subsystem (IMS) Application

Level Gateway (IMS-ALG) - IMS Access Gateway (IMS-AGW) interface: Procedures descriptions. http://www.3gpp.org/DynaReport/23334.htm, March 2015. [Online; accessed 16-april-2014].

[34] Patrick Park. *Voice over IP Security*. Cisco Press, 2009.

[35] Nokia Networks. Voice evolution: Leveraging Voice over Wi-Fi. http://networks.nokia.com/sites/default/files/document/nokia_voice_evolution-leveraging_voice_over_wi-fi_webinar_presentation.pdf, 2014. [Online; accessed 16-April-2015].

[36] Tammy Noergaard. *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*. Newnes, second edition, 2013.

[37] Enea. Enea® Linux Real-Time Guide. http://www.enea.com/Embedded-hub/whitepapers/white-papers/Linux-Real-Time-Guide/, 2014. [Online; accessed 23-April-2015].

[38] Enea. LWRT. http://www.enea.com/solutions/rtos/LWRT/, 2014. [Online; accessed 23-April-2015].

[39] Intel. Intel® Xeon® Processor E5-2665. http://ark.intel.com/products/64597/Intel-Xeon-Processor-E5-2665-20M-Cache-2_40-GHz-8_00-GTs-Intel-QPI. [Online; accessed 23-April-2015].

[40] Codenomicon. The Heartbleed Bug. http://heartbleed.com/, 2014. [Online; accessed 22-October-2014].

[41] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson. Performance Comparison of the AES Submissions. https://www.schneier.com/paper-aes-performance.pdf, February 1999. [Online; accessed 13-April-2015].

A. APPENDIX

Scaling the DSP encryption performance Figure 6.4 to one tenth of the connections to reflect the same amount of relative load as the virtual platform is shown below and it becomes clear that the application behaves very similarly on both platforms when the performance difference is taken into account. The trend of the graph in Figure A.1 is very similar as the virtual application performance graph in Figure 6.6.

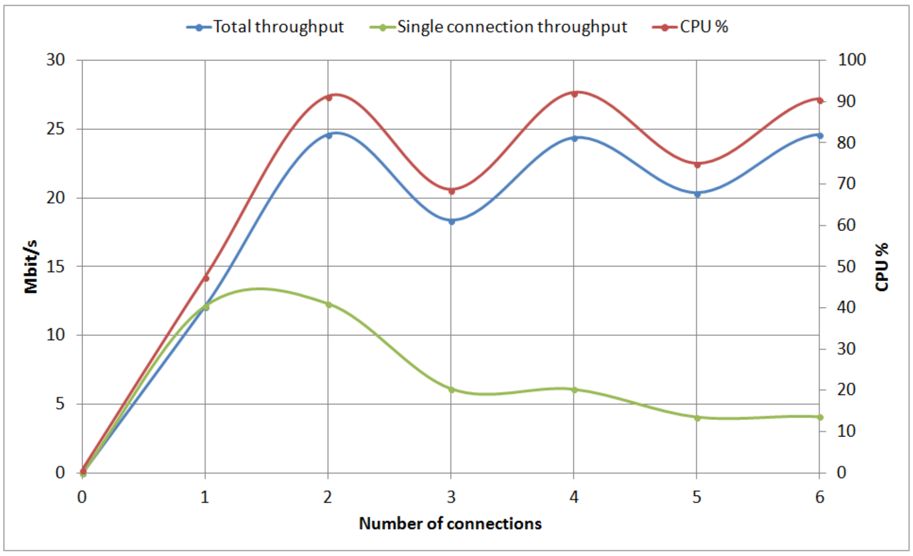

Figure A.1: Same as DSP performance graph in Figure 6.4, but scaled to 6 connections.

Scaling Figure 6.3 to one sixth of the connections to reflect the same relative load as the virtual platform in the case without encryption reveals the similarities. Trends in Figure A.2 and Figure 6.5 are very similar, as the application behaviour between platforms is very comparable, when the performance difference is taken into account.

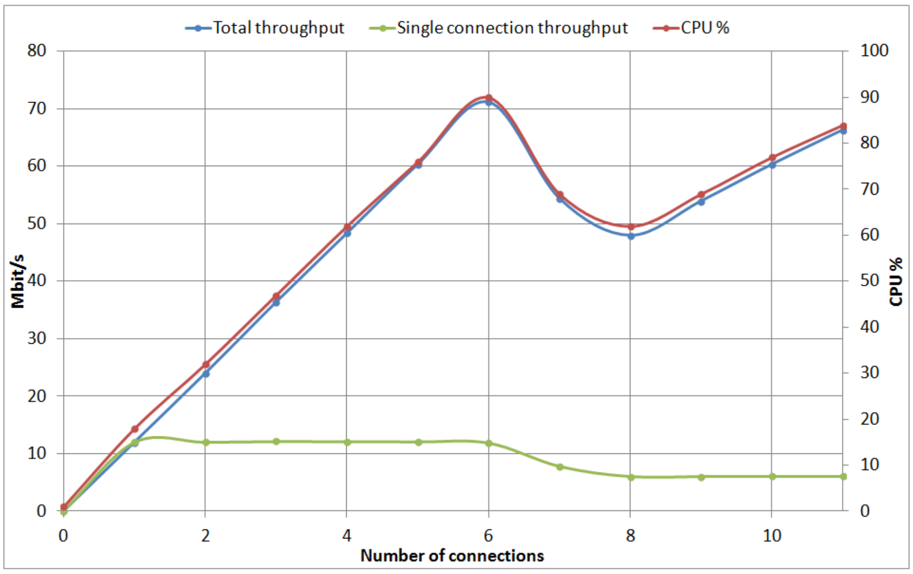

Figure A.2: Same as DSP performance graph without encryption in Figure 6.3, but scaled to 11 connections.