Study of mechanisms ensuring service continuity for IKEv2 and IPsec protocols

Daniel Palomares

THÈSE DE DOCTORAT CONJOINT TÉLÉCOM SUDPARIS
et L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité:

École doctorale: Informatique, Télécommunications et Électronique de Paris

Présentée par
Daniel Palomares Velasquez

Pour obtenir le grade de
Docteur de Télécom SudParis

Étude de Mécanismes Assurant la Continuité de Service de Protocoles IKEv2 et IPsec

Soutenue le 14 Novembre, 2013 No. de thèse: 2013TELE0025

Devant le jury composé par:

BONNIN Jean-Marie, Professeur HDR à Télécom Bretagne	Rapporteur
CARLE Georg, Professeur à Technische Universität München	Rapporteur
PUJOLLE Guy, Professeur à l'Université Pierre et Marie Curie - LIP6	Examinateur
COMBES Jean-Michel, Docteur et Ingénieur à Orange Labs de France Télécom R&D	Examinateur
MIGAULT Daniel, Docteur et Ingénieur à Orange Labs de France Télécom R&D	Examinateur
LAURENT Maryline, Professeur HDR à Télécom SudParis - Departement RST	Directrice de Thèse

PhD THESIS AGREEMENT TÉLÉCOM SUDPARIS
AND UNIVERSITY PIERRE ET MARIE CURIE

Specialty:

Doctoral School: Informatique, Télécommunications et Électronique de Paris

PhD Thesis Dissertation by
Daniel Palomares Velasquez

To obtain the degree of:
Doctor of Télécom SudParis

Study of Mechanisms Ensuring Service Continuity for IKEv2 and IPsec Protocols

Defense date: November 14th, 2013 No. of thesis: 2013TELE0025

Members of the jury:

BONNIN Jean-Marie, Professor HDR at Télécom Bretagne	Reporter
CARLE Georg, Professor at Technische Universität München	Reporter
PUJOLLE Guy, Professor at University Pierre et Marie Curie - LIP6	Examiner
COMBES Jean-Michel, Doctor and Engineer at Orange Labs of France Télécom R&D	Examiner
MIGAULT Daniel, Doctor and Engineer at Orange Labs of France Télécom R&D	Examiner
LAURENT Maryline, Professor HDR at Télécom SudParis - Departement RST	Thesis Director

Acknowledgments

First of all I would like to render thanks to every single person that help me out through all my career. Specially my parents, which are my biggest inspiration.

The first steps of this thesis were done thanks to Professor Maryline Laurent, my thesis director. I can only say thanks for her support and patience as well as for the time she has consecrate to this research.

Second, I would not be able to make this investigation without the help and supervision of Daniel Migault. I admire his capability, creativity, professionalism and enthusiasm. Throughout these years, I have learned a lot from him and I am truly grateful.

I also had the opportunity to conduct several academic projects and internships. They all have been a big source of apprenticeship to me.

I am also grateful towards Vincent Barriac, which I thank you for his investment in producing excellent quality results concerning the last part of this thesis.

These persons also helped me throughout the achievement of my thesis: Wolfgang Velasquez, Martin Willi, Jean-Michel Combes, Hendrik Hendrik, Emmanuel Herbert, Aurelien Wally, Jose Daniel Gomes, Ana Maria.

Being part of Orange has been a truly remarkable experience for my professional career, and I Thank You All.

Daniel Palomares.
September 2013.

Abstract

During 2012, the global mobile traffic represented 70% more than 2011. The arrival of the 4G technology introduced 19 times more traffic than non-4G sessions, and in 2013 the number of mobile-connected to the Internet exceeded the number of human beings on earth. This scenario introduces great pressure towards the Internet service providers (ISPs), which are called to ensure access to the network and maintain its QoS. At short/middle term, operators will relay on alternative access networks in order to maintain the same performance characteristics. Thus, the traffic of the clients might be offloaded from RANs to some other available access networks. However, the same security level is not ensured by those wireless access networks. Femtocells, WiFi or WiMAX (among other wireless technologies), must rely on some mechanism to secure the communications and avoid untrusted environments.

Operators are mainly using IPsec to extend a security domain over untrusted networks. This introduces new challenges in terms of performance and connectivity for IPsec. This thesis concentrates on the study of the mechanism considering to improve the IPsec protocol in terms of continuity of service.

The continuity of service, also known as resilience, becomes crucial when offloading the traffic from RANs to other access networks. This is why we first concentrate our effort in defining the protocols ensuring an IP communication: IKEv2 and IPsec. Then, we present a detailed study of the parameters needed to keep a VPN session alive, and we demonstrate that it is possible to dynamically manage a VPN session between different gateways. Some of the reasons that justify the management of VPN sessions is to provide high availability, load sharing or load balancing features for IPsec connections. These mechanisms increase the continuity of service of IPsec-based communication. For example, if for some reason a failure occurs to a security gateway, the ISP should be able to overcome this situation and to provide mechanisms to ensure continuity of service to its clients.

Some new mechanisms have recently been implemented to provide High Availability over IPsec. The open source VPN project, StrongSwan, implemented a mechanism called ClusterIP in order to create a cluster of IPsec gateways. We merged ClusterIP with our own developments in order to define two architectures: High Availability and Context Management over Mono-LAN and Multi-LAN environments. We called Mono-LAN those architectures where the cluster of security gateways is configured under a single IP address, whereas Multi-LAN concerns those architectures where different security gateways are configured with different IP addresses.

Performance measurements throughout the thesis show that transferring a VPN session between different gateways avoids re-authentication delays and reduce the amount of CPU consumption and calculation of cryptographic material. From an ISP point of view, this could be used to avoid overloaded gateways, redistribution of the load, better network performances,

improvements of the QoS, etc. The idea is to allow a peer to enjoy the continuity of a service while maintaining the same security level that it was initially proposed.

Contents

Bibliography

List of Figures

List of Tables

Chapter 1

Introduction

The Internet Service Providers (ISPs) are facing big challenges in order to manage the amount of mobile data traffic. During 2012, the global mobile traffic represented 70% more than 2011. Actually a 4G connection generated, in average, 19 times more traffic than a non-4G session. Moreover, in 2013 the number of mobile-connected devices will exceed the number of human beings, making the aggregate smartphone traffic in 2017 be 19 times greater than it is today [1].

Radio Access Networks (RAN) will not be able to face the exponential growth of such traffic. Thus, in order to avoid overloaded networks, operators would have to offload RAN traffic towards WLANs. While offloading, ISPs must provide the same Quality of Service (QoS) and equivalent trust level. Actually, RAN networks are considered trusted domains whereas WLANs networks are considered untrusted networks. However, one solution that provides a similar trust level over WLANs networks are the VPNs (Virtual Private Networks).

One of the protocols that aims to protect IP traffic is IPsec, which is largely deployed as IPsec-based VPNs. These VPNs are designed to extend a trusted domain over an untrusted network. A VPN requires an End-User (EU) and the access network to agree on some security parameters in order to secure their IP traffic. This is done by establishing security associations (SA) between a **Security Gateway** (SG) and the EU. This may result in a large cluster of SGs managed by the ISPs at the same time.

The objective of this thesis is to allow IPsec VPN infrastructures to provide high QoS to EUs as well as to ensure continuity of service. More specifically, we want the VPN connections to be resilient to failures. This is what we call *High Availability* (HA) throughout this thesis. We also want that ISPs can easily manage the load due to the VPN traffic of their clients (this is what we call *Traffic Management* or *Context Management*). As such, when a multinode VPN platform is being used by the ISP, it is able to transfer a VPN connection from one SG to another, introducing the concept of *context transfer*. Thus, when a VPN session is transferred, this means

that all the parameters concerning that session are transferred as well. In fact, there are some scenarios where the possibility to transfer a VPN session from one SG to another has beneficial effects. Some use cases involve load-balancing of VPN traffic among different SGs, handover between different access networks or improvements of fail-over clusters with High Availability (HA) features.

Throughout this chapter, we present some architectures of interest and use cases in section 1.1. Then, we describe the VPN Service and introduce some technical challenges concerning the transfer of the security contexts. We also introduce two main architectures called Mono-LAN and Multi-LAN in section 1.2. Following section 1.3, presents the theoretical positioning of Context Transfer face to Mobility operation. Section 1.4 positions our investigations compared to some related works. Later on in section 1.5, we summarize our contributions to the community during this thesis, and finally, section 1.6 describes the organization of the manuscript.

1.1 Architectures of Interest

The goal of this section is to introduce some architectures of interest and scenarios where the IKEv2/IPsec context transfer may be used in real topologies. First, we explain the advantages of transferring the IKEv2/IPsec parameters and how we can use it to build a cluster of IPsec-based VPN service. Then, we present how the context transfer may improve EUs experience in *offload* scenarios.

1.1.1 Clustering IPsec Security Gateways

This architecture aims to create a cluster of IPsec Security Gateways. IPsec [2] is a layer 3 protocol aiming to protect IP communications. An End-User (EU) accessing the network must establish a VPN connection with a SG in order to gain access to some services. The goal of the cluster is to render the IPsec services highly available. The traffic between the EU and the cluster is protected with IPsec and the session is authenticated using Internet Key Exchange protocol (IKEv2). Our goal is to create a cluster of SGs aiming to improve the EU experience while protecting the communication with IKEv2/IPsec suite.

Figure 1.1 represents a cluster of SGs. It shows an EU establishing a VPN session with the cluster. For some reason (e.g. an overloaded SG, a failure on the responsible SG, latency, etc.), the SG must transfer its session to another SG within the cluster in order to ensure continuity of the VPN service. By doing so, the EU's experience is improved because this represents less interruption of the communication, as the EU does not need to proceed a time consuming operation to re-establish a new VPN session from scratch. However, the session transfer requires

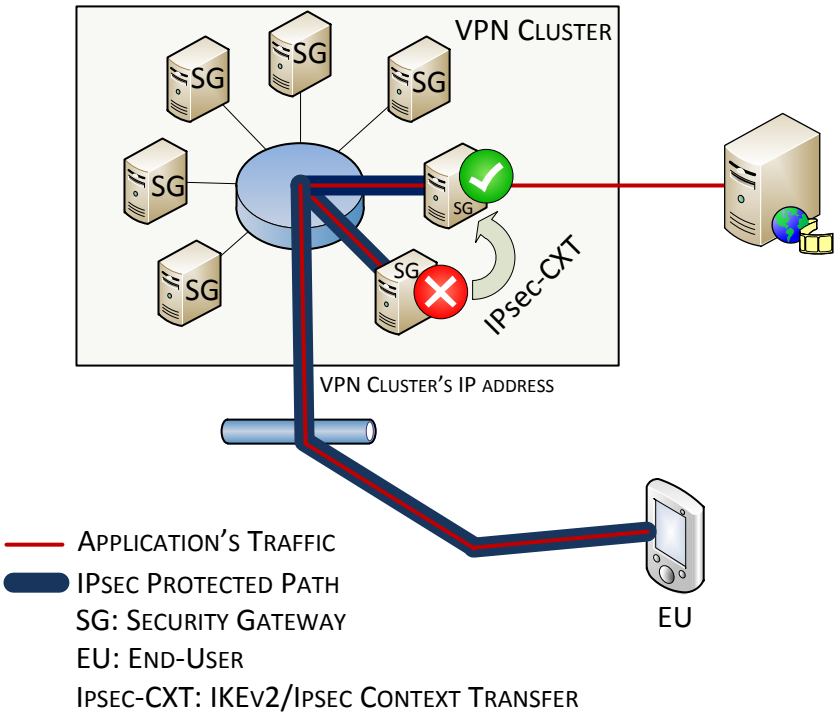


Figure 1.1: Clustering IPsec security gateways

also transferring the IKEv2/IPsec context from one Security Gateway to another. Two situations may come up; whether the SGs share the same IP address or have two different IP addresses. These situations are next referred to as Mono-LAN or Multi-LAN environments respectively (see section 1.2).

1.1.2 Offload Architectures

The mechanism called *offload* consists in switching the End-Users (EUs) from overloaded Radio Access Networks (RANs) towards some alternate wireless networks (e.g. WLAN like for example WiFi). These networks are more likely to be untrusted (when attaching other operator's access points), unreliable and prone to not handle mobility operations. As a result, some operations like mobility, multihoming or security must be handled by the terminal itself (see [3]).

Once offloading towards the WLANs, operators need to maintain the same level of security as in RANs. For this reason, applying security at the radio layer only (L2) might not be sufficient since the WLAN access point is considered as untrusted when it does not belong to the same service provider. This leads the limitation of offloading only between the same operator's access points. Another solution might be to secure at a higher layer, for example, at the transport layer or above (L4 or more). This means that applications have to be security-aware. Unfortunately,

the applications in the market are reluctant to modify their source code for applying some sort of security (e.g. TLS or DTLS), as the EUs experience might decrease due to additional overhead, bandwidth consumption, additional certificate popups and additional computation. Furthermore, operators might favor to secure their communication with EUs at the IP layer (L3), as IPsec, to secure IP communications. IPsec not only secures at the IP layer but also secure the layers above.

Figures 1.2 and 1.3 show two cases of interest representing some EUs being offloaded from RAN to WLAN. The first use case illustrates an EU attaching a cluster within a same administrative domain, whereas the second use case illustrates an EU moving between two VPN clusters placed in different administrative domains.

IKEv2/IPsec Context Transfer within the Same Administrative Domain

Figure 1.2 represents the first use case. An EU is offloaded from its RAN to a WLAN (WiFi). The peer needs to gain access to the multimedia server placed behind the cluster. Once the EU attaches the WLAN Access Point (AP), it obtains its IP address, which is used to establish a VPN tunnel towards the cluster. At this point, the EU gains access to the multimedia server in a secure manner. Since secure attachment is provided at the IP layer with the VPN cluster located in the ISP CORE network, the communication remains trusted even though WLAN APs may be untrusted.

There are some advantages whenever the operator may decide to transfer the VPN session from one SG to another SG within the cluster by implementing IKEv2/IPsec context transfer. For example, a VPN service provider may want to offer High Availability capabilities, support self-organization of the cluster, avoid overload SGs, improve bandwidth, reduce latency, among other features. However, if this context transfer happens between the same administrative domain where the SGs are configured with a single IP address (IP_AD_1 in fig. 1.2), the EU is concerned by any signaling operations. Context transfer is transparent for EUs, as the IP addresses of both extremities of the VPN tunnel remain the same.

IKEv2/IPsec Context Transfer between Different Administrative Domains

In contrast with the precedent use case, figure 1.3 represents the context transfer of the IKEv2/IPsec context between VPN clusters within different administrative domains. When this happens, the IP address of the VPN tunnel changes (from IP_AD_1 to IP_AD_2). Throughout this thesis, we call this kind of scenario Multi-LAN environments (see following section 1.2.1). This situation has an impact over the EU, because some IKEv2 signaling must take place in order to update the VPN session associated to the new IP address. The advantage of the context transfer between different administrative domains is that the EU avoids re-authenticating when moving from AD_1 to AD_2. Additionally, the EU continues to benefit from the features of a VPN cluster (e.g. load balancing, high availability, load sharing, etc.).

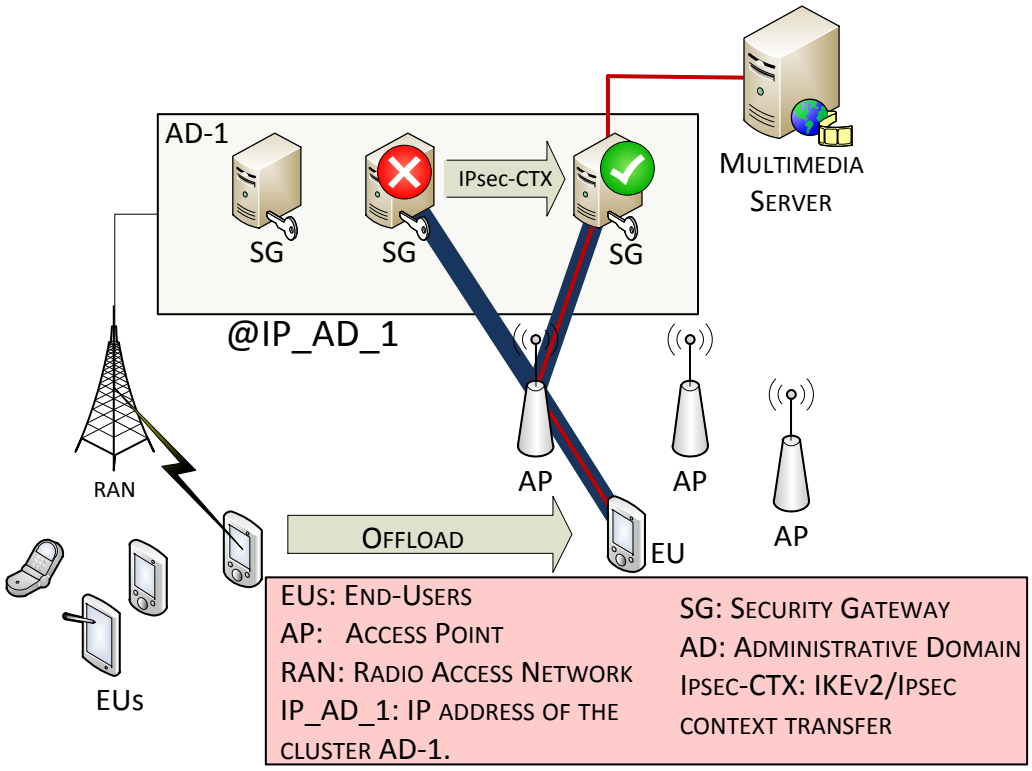


Figure 1.2: Offload Architecture: Context transfer within a same administrative domain

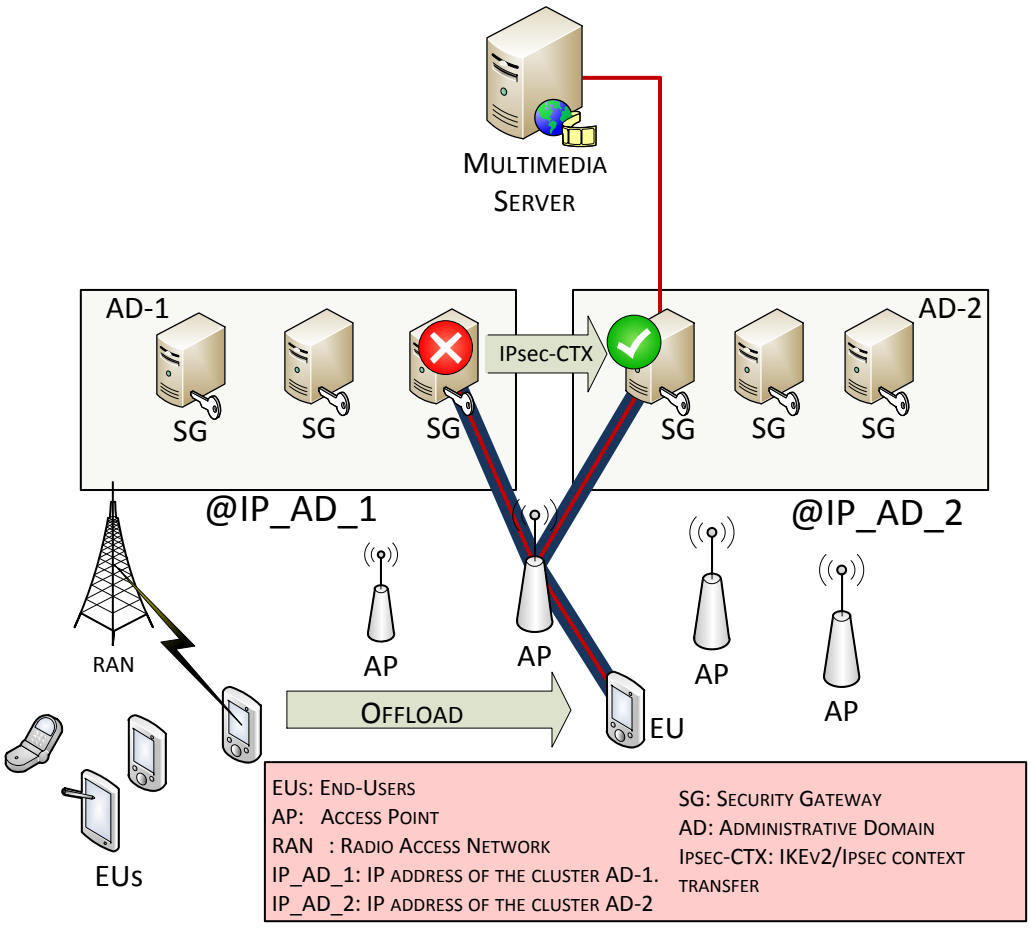


Figure 1.3: Offload Architecture: Context transfer between two administrative domains

1.2 VPN Service description and technical challenges

1.2.1 High Availability of IPsec-based VPNs

A VPN service that offers *High Availability* (HA) features will improve EU's experience when a failure occurs during an active session of an IPsec-based VPN. Indeed, the technical challenges to offer *High Availability* capabilities involves **detection** of a SG that does not work anymore, and letting a new SG to take responsibility of a previously established VPN session.

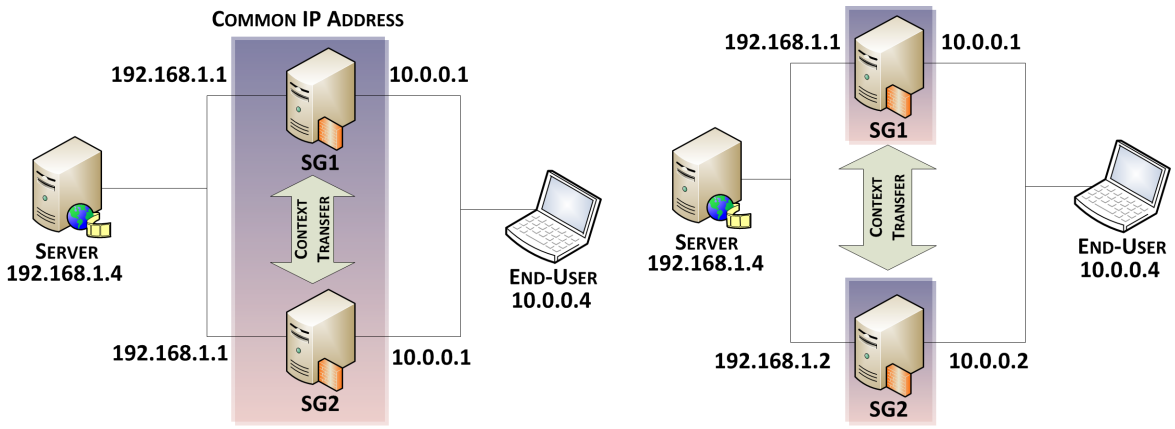
When an EU establishes a VPN with a SG, some cryptographic material and keys are derived between them. When the SG fails, we do not want the EU to re-negotiate all these parameters with another stand-by SG which is supposed to take responsibility of the VPN connection. Thus, the cryptographic material and keys must be shared between the failing SG and the stand-by SG. These parameters are referred to as the IKEv2/IPsec context and are explained in section 2.2.

We will consider the following architectures:

- **Mono LAN:** This architecture consists in a set of SGs that own the same IP address. As represented in figure 1.4a, both SGs share the same IP. However, unless some special mechanism is used, both SGs can not own the same IP address at the same time, due to duplicated IP conflicts (this is discussed in detail throughout the thesis). In this case, the EU does not need to update its IP attachment address when its connection moves from SG1 towards SG2, and the IKEv2/IPsec context is also transferred between both SGs.

- **Multi LAN:** The second architecture consists in a set of SGs that own different IP addresses. Figure 1.4b represents a Multi-LAN architecture, where SG1 and SG2 have different IP addresses. In this case, when a context transfer takes place, the EU is impacted because it must update its VPN's destination IP address (i.e. one of the extremities of the tunnel). The EU can face this issue by performing specific IKEv2 exchanges like MOBIKE or REDIRECT extensions (see sections 2.6 and 2.7).

The biggest challenge to overcome a failure and to offer HA features concerns the synchronization of counters of the IKEv2/IPsec context. These counters control every incoming/outgoing IP packet protected with IPsec, preventing replay or unauthorized re-injection of previously processed IPsec traffic. These counters are referred together as IKEv2/IPsec counters. They must stay synchronized on both extremities when transferring a VPN session between different SGs. Ensuring their synchronization is a challenge that is addressed throughout this thesis.



(a) Mono-LAN Architecture (b) Multi-LAN Architecture

Figure 1.4: IKEv2/IPsec context transfer scenarios

1.2.2 Traffic Management of IPsec-based VPNs

Transferring a VPN connection from one SG to another allows an EU to change its point of attachment towards the access network. The main challenge of transferring the VPN traffic concerning an IPsec connection is to enable a SG to transfer all the cryptographic material and derived keys from one SG to another. Traffic management might be confusing with HA feature. However, they differ in the fact that **traffic management does not involve failure detection** of a SG, while the HA features do detect this situation. As such, traffic management can be triggered whenever it is desirable, whereas HA capabilities are automatically launched once a SG fails.

As represented in figure 1.4a, the Mono-LAN architecture illustrates the case where both SGs have the same IP address. During a transfer of an IKEv2/IPsec context in Mono-LAN might have no impact to EUs. The point of attachment remains virtually the same, even though it is a new SG with the same IP address which ensures the communication. On the other hand, as represented in figure 1.4b, the Multi-LAN architecture illustrates the case where two SGs have different IP addresses. Thus, the EU is impacted because his point of attachment has changed too. These modifications can be done using MOBIKE or REDIRECT extensions. Note that the Multi-LAN scenario needs transferring IKEv2/IPsec context, which is not feasible with current standards.

Finally, table 1.1 summarizes the solution that addresses each scenario. For example, when performing traffic management under a Mono-LAN architecture, we need to perform Context Transfer in order to maintain the VPN session.

	Mono-LAN	**Multi-LAN**
High Availability HA	ClusterIP	Context Transfer + IP Mobility + HA Module
Traffic Management	Context Transfer	Context Transfer + IP Mobility

Table 1.1: Solutions for each scenario

1.3 Context Transfer is not designed for Mobility

For clarity, this section shows how IKEv2/IPsec context transfer differs from protocols designed for mobility operations. The confusion between IKEv2/IPsec context transfer and mobility comes from the fact that a context transfer makes possible that a given IPsec session between an EU and a Security Gateway can be "moved" to another Security Gateway; whereas a mobility operation occurs when the EU changes its IP address. On the other hand, mobility use cases for Mobile IP (MIP) or MOBIKE (mobility extension for IKEv2), consider an EU attached to a Home Agent or a Security Gateway respectively. IP datagrams are tunneled to the EU. As a result of mobility, the outer IP address of the EU changes and needs to be updated in the Security Gateway or the Home Agent.

There are two available mechanisms to handle mobility and redirection of an IKEv2/IPsec context:

- **MOBIKE** [3] is a mobility and multihoming extension for IKEv2. MOBIKE considers the following scenarios when an EU is connected to a SG using tunnel mode: first, if one of the peer changes its IP address, it can advertise the other node that its IP address has changed. This means that the outer IP address of the tunnel is updated. Note that with MOBIKE, a peer can advertise that its IP address has changed as well as it can be aware of mobility from the other extremity of the tunnel. The second scenario is multihoming. One of the peers can advertise the other peer of some alternate IP addresses where it is reachable if ever needed. These alternate IP addresses should be used in case the running IP address is not reachable anymore.

- **REDIRECT** [4] is also an extension for IKEv2. It has been designed to redirect an IKEv2/IPsec session from one Security Gateway to another Security Gateway. The SG sends a REDIRECT message to the EU, indicating the new SG to attach. When the EU receives this message, it breaks the IKEv2/IPsec Security Associations established with the currently active SG and re-establishes/renegotiates all Security Associations with the new SG. Note that the EU is forced to renegotiate all the security parameters from scratch when being redirected to another SG. This may impact EU experience due to network delays while establishing a new VPN towards another SG. REDIRECT may happen when establishing of a VPN tunnel or during an active VPN session. On the

other hand, REDIRECT mechanism does not consider the transfer of an IKEv2/IPsec context, which could actually ensure better continuity of service and improve EU's QoS.

Now let us point out the differences between the context transfer and mobility operations:

1. With Mobility, the communication is established between two entities, the EU and the SG (or the Home Agent). These two entities remain the same before and after mobility occurs. Only the IP address of the EU is changed. On the other hand, with IKEv2/IPsec context transfer, the two entities before the context transfer and after the context transfer are different. In our case the SG is a different piece of hardware (device).

2. Mobility is only focused on the IP address, whereas IKEv2/IPsec context transfer carries all necessary security parameters of the IPsec session. This results from the fact that with mobility the entities remain the same, so both peers are aware of the session parameters.

In this section we mainly pointed out the differences between IKEv2/IPsec context transfer and mobility, as different protocols addressing different issues. This also means that these cases can be used together in a complementary manner. For example, when the IKEv2/IPsec context transfer takes place between two SGs that own different IP addresses (this is what we call a Multi-LAN architecture), then we can use mobility operations together with context transfer. Both mechanisms can be used as complementary solution, allowing to update the outer IP address of the tunnel as well as maintaining the IKEv2 and IPsec information previously negotiated. In summary, performing **mobility and context transfer** ensures a fluid continuity of service for a mobile node that changes it attachment IP address between different SGs.

1.4 Related work and position of our work

A few publications and works have already been published on IKEv2/IPsec context transfer, and to our knowledge, most of them considered a Mobile IP environment. In fact, when using Mobile IP, IPsec may be used to protect the tunnel between the Mobile Routers and the Home Agents and to protect the signaling between some access router and the EU.

1.4.1 Position towards theoretical proposed architectures

Georgiades et al. exposed a theoretical case of study in [5], where IPsec context transfer optimizes the transfer of some security context from a given device to another without renegotiating from scratch all the IPsec parameters. Georgiades et al. also introduced the concepts of homogeneous and heterogeneous security context transfer. This is actually the same analysis we arise when

describing a context transfer between the same administrative domains or between different administrative domains. Some authentication protocols (e.g. RADIUS, Diameter) are discussed throughout the article in order to determine the potential parameters during a security context transfer, but no details concerning the IPsec context are given. However, this study does not include any performance test or real implementation nor any simulation.

Similarly, [6] describes also a theoretical study proposition for an aeronautical use case, where an IKEv2/IPsec context is transferred between a Mobile Router and the Home Agent by using Context Transfer Protocol (CXTP [7]). They show that using IKEv2/IPsec context transfer saves almost 10 times the overhead produced by IKEv2/IPsec tunnel establishment. It presents the theoretical case where a Mobile Router switches its IPsec SAs from one Home Agent to another Home Agent, and presented IKEv2/IPsec context transfer as one way to optimize this inter Home Agent migration. The context transfer can be either triggered by the Mobile Router or by the Home Agent. Finally, a MOBIKE exchange is performed in order to update the IPsec database on both the MR and the new Home Agent, so the communication within the tunnel is maintained. However, the overhead calculation made in [6] does not take into account the additional delays during the CTD (context transfer data) exchanges between the old Home Agent and the new Home Agent. On the other hand, [6] does not give details of IKEv2/IPsec context parameters. Nevertheless, they explain the IKEv2 exchanges needed to establish an IKEv2/IPsec session. No implementation or performance test is conducted neither, although is considered as future work. In contrast, our investigation counts with several testbeds and performance tests.

[6] and [5] both introduce different theoretical frameworks for IPsec context transfer in mobility environments (Mobile IP - MIP). In fact, they use MIP for session mobility and MOBIKE extension for IPsec mobility. The Multi-LAN architecture addresses similar mobility use cases. However, we do not use MIP for the terminal mobility. Instead, we only based our mobility operations by using the MOBIKE extension of IKEv2. We believe that using a single protocol (e.g IKEv2) for session mobility, reduces the complexity of the architecture, managed by operational teams and thus favors its deployment. Furthermore, our work is not motivated by mobility but by High Availability and management of SG clusters. Mobility concerns one session whereas High Availability concerns all IPsec sessions of a given SG. This clearly rises scalability issues when all the IPsec contexts have to be transferred from one SG to the other. To address this issue, we define different manners to synchronize different IPsec databases of different SG and we define different strategies for transferring these contexts. In addition, we do not only define the architectures but we also implement and test the transfer of the IPsec context (see chapter 4).

Allard et al. [8] and Ayaz et al. [6] addressed the problem of uniqueness of the Security Parameter Index (SPI) of IPsec Security Associations. The SPI is used to uniquely identify the

corresponding security association of a tunnel. However, a collision of SPIs could happen when transferring IKEv2/IPsec contexts between different devices. Allard et al. psopose MOBIKE in order to avoid SPI collisions. In contrast, [6] uses HELLO messages (see [9]) to prevent SPI collisions before the context transfer takes place. In order to address this issue in Multi-LAN environments, we use the solution described in [8], as we base our mobility operation solution by using MOBIKE. Allard et al. in [8] identified all the parameters for both IKEv1/IPsec and IKEv2/IPsec contexts. However, the transfer of context was implemented for MIP architectures and not High Availability ones, which involve different constraints as presented in section 1.3. Following section 1.4.2 positions our work from an implementation point of view with [8].

1.4.2 Position towards existing implementation

Allard et al. in [8, 10–14] are the only pieces of work that mention the existence of an implementation concerning IPsec context transfer. Allard et al. not only proposed the CXTP protocol to transfer an IKE/IPsec context as in [6], but they also considered other use cases as the *Protocol for Carrying Authentication for Network Access* (PANA) and Mobile IP (MIP). In [8], chapters 9, 11, 12 and 13 are dedicated to present the PANA context transfer simulation based on a software called OMNet++.

Considering the similarity of our investigation with the work done by Fabien Allard, in the remaining of this section we next position our work against his thesis [8]:

Throughout this thesis, we concentrated on the usage of the version 2 of the Internet Key Exchange (IKE) protocol as the mechanism to perform authentication between a node and a SG. All of the architectures in [5,6,8,10–14], are using MOBIKE to update the IPsec databases after the context tranfer occurs. MOBIKE is an extension of IKEv2 and there are no equivalence for IKEv1. Even though IPsec context transfer for IKEv1 has been implemented by Fabien Allard, the proposed architectures had never been evaluated with a real implementation. In order to evaluate this architecture, a new implementation based on IKEv2 is needed. In addition, IKEv2 proposes multiple extensions that our architectures are using: management of IKEv2/IPsec sessions (i.e. REDIRECT), mobility and multihoming operations (i.e. MOBIKE) and High Availability protocols [15].

Allard et al. implemented the IKEv1/IPsec context transfer on *racoon* implementation. *Racoon* is actually deprecated and there is no upgrade to IKEv2. We choose to implement the context transfer on *strongswan* [16], the reference open source implementation for IKEv1 and IKEv2, and we could hardly reuse the code developed for *racoon* implementation. In addition, Allard et al. in [8, 11] use an implementation called PF_KEY to manipulate the IPsec related kernel information. PF_KEY originates from BSD, and is implemented in Linux as well. However, it is now deprecated and was replaced by an implementation called XFRM,

which is newer and more robust. We based our implementation on XFRM, which actually allows updating sequence number values of IPsec Security Associations. We managed to separate each parameter of the security association within the implementation in order to update these counters (for IKE and IPsec), whereas such achievement ins not possible with PF_KEY.

Allard et al. tested their implementation using an UDP traffic generator. This traffic generator has been configured to reproduce UDP packets every 50ms within a single VPN established session. We consider this experimentation is a proof of concept, however, as mentioned by Allard et al., under heavy traffic situations, more work is needed to handle IKEv2/IPsec counter synchronization. In our case, we tested our implementation under heavy load situations, which give us a clear response of the impact against upper layers. Additionally, we tested how IKEv2/IPsec context transfer impacts real time services through network measurements as well as Quality of Services (using POLQA [17]) for audio streaming services. In contrast with UDP traffic generator, we tested our architecture with different transport layers like TCP/HLS, UDP/RTSP; measuring the impact at the application layer. Using QoS, we prove that our solution has no major impact on End-User's services.

1.4.3 Detailed position towards High Availability vs. Mobility scenarios

In an architectural point of view, our work differs from previous work in the use cases we address. The scenarios that motivated our work are not mobility but High Availability and Cluster management. In a Mobile IP (MIP) environment, a Mobile Router (MR) is transferred from one Home Agent to another Home Agent, and the IKEv2/IPsec context is transferred between Home Agents with the purpose to optimize the handover and reduce latency. On the other hand, our work considers two different scenarios, and both involve clusters of Security Gateways.

The first scenario considers a cluster of Security Gateways. All of them are seen by the EU as a single IP address. Motivation for moving a session from one SG to the other is to manage the cluster and to provide High Availability features. The difference with Mobile IP environments, is that no interaction is required between the EU and the Security Gateway during the IKEv2/IPsec context transfer (the transfer is transparent to EUs). Also, by providing High Availability, each SG within the cluster is associated to a standby node that takes in charge the traffic in case of failover. In summary, the main differences compared to mobility operations are explained as follows:

1. First, failovers cannot be prepared in advance like in mobility operations. Second, all communications are transferred in a High Availability scenario, instead of one single tunnel during mobility. These two constraints impose to synchronize IPsec databases between the active node and the standby node within the cluster. This shared database requires to be

updated for every IKEv2 or IPsec packet, since any inbound or outbound packet updates IPsec or IKE counters. With heavy load this is hardly possible, as a result we define different strategies according to the load the Security Gateways have to deal with.

2. We define a synchronization timer for IKEv2 and IPsec databases. This timer can be dynamically assigned according to the load,the more loaded the active node is, the longer the timer should be. Once the duration of the timer has been defined, the IPsec and IKEv2 databases are synchronized periodically based on the timer policy. During a failover event, the IPsec standard allows to increase the IPsec database counters in order to resynchronize stale values once a context transfer is performed. In contrast, if the timer cannot be increased (e.g. heavily loaded SGs scenarios), then the active node must perform additional IKE exchanges in order to resynchronize IKE_SA's counters (see chapter 4). The reason why IKE_SAs synchronization gets higher priority compared to IPsec_SAs, is that an IPsec_SA can be re-synchronized using different mechanisms like REKEY, creation of a new IKE_SA or eventually resynchronization using the IKEv2 channel. However, if desynchronization occurs with the IKEv2 counters, then synchronization through IKE exchanges is the only mechanism that makes possible IKEv2 counter resynchronization. Currently we are not aware of any implementation of this protocol (see [15]).

3. When loads keep on increasing, then counters cannot be synchronized anymore. To avoid a re-negotiation of the IKEv2 channel, the EU must implement the synchronization protocol.

The second scenario we considered is a transfer between clusters that have different IP addresses. In this case, interaction between the EU and the clusters is necessary. However, transfer from one cluster to the other is only performed using IKEv2 signaling. In other words, we are not using Mobile IP signaling to perform mobility.

Finally, we present a solution to offer IPsec High Availability features. For this reason, we focus on a mechanism named ClusterIP, consisting in creating IP-based clusters with no dedicated hardware. ClusterIP allows to implement High Availability features and ensure connectivity during failovers of the SGs. The goal of ClusterIP is to share a common IP address with the purpose to build a cluster of machines in order to spread the load of incoming/outgoing IP packets. ClusterIP is based on firewalling rules under Linux environments. However, in order to implement ClusterIP under IKEv2/IPsec environments, we had to use the XFRM implementation that allows to get and set the IPsec sequence numbers.

1.5 Contributions

Our first contribution concentrates on identifying the elements composing an IKEv2/IPsec context (i.e. all the cryptographic material, derived keys and other parameters that are

negotiated during the establishment of a VPN tunnel).

A second contribution, implements two new functions called **GET** and **PUT**, aiming to extract and inject an IKEv2/IPsec context on a real IPsec stack. This work is done based on the VPN open source solution called *strongSwan* (v4.5.0) and Linux (Ubuntu 11.04). Our developments are published as open source projects and can be used by the research community. Additionally, we establish an experimental platform, and we perform intensive performance tests. For instance, we conduct some tests with heavily loaded SGs (several VPN tunnels at the same time), demonstrating how smooth can be an IKEv2/IPsec context transfer, and how interesting is this approach for real deployment by operators.

Our third contribution is related to the Mono-LAN environment and gives an extensive performance comparison between an existing ClusterIP solution and our context transfer approach. Once again, we performed some performance tests with real trafic (e.g. streaming with TCP or UDP), demonstrating the feasability of the proposed solution.

We also approached the IKEv2/IPsec context transfer implementation under Multi-LAN environments. We present a theoretical framework to perform context transfer between SGs with different IP addresses.

Throughout this thesis, we faced some difficulties for addressing an appropriate solution. The first obstacle concerned the complexity and understanding of IPsec together with IKEv2. Both protocols are defined in different standards and involve a massive list of parameters that are negotiated in order to establish a VPN tunnel. However, as in any research project, this is part of an exhaustive study. Then, further investigations lead us to face some development challenges. Getting inside the source code of strongSwan is a complex task. Understanding the features of this VPN solution represented one of the main difficulties throughout this thesis. The following difficulties concerned mainly networking and security network issues. In addition, our validation tests are also consider as a proof of concept. However, we insisted in running performance tests and measuring the impact on EU's experience for each scenario studied.

These investigations were published as scientific papers at different conferences: The International Conference on Secure Networking and Applications [18] (ICSNA 2013), The Third International Conference on Communications and Information Technology [19] (ICCIT 2013), the 8th International Conference on Availability, Reliability and Security [20] (ARES2013) and Proceedings of IEEE Global Telecommunications Conference - Communication and Information System Security [21] (GLOBECOM '12) .

1.6 Thesis Organization

The remainder of the manuscript is composed of six chapters. Chapter 2 introduces all the basic knowledge needed for a good understanding of the document. The description includes IKEv2/IPsec protocols, and also introduces REDIRECT, MOBIKE and ClusterIP mechanisms.

Chapter 3 identifies the elements to be included in the IKEv2/IPsec context. We present our initial results of GET and PUT functions, by illustrating the time it takes to recover an IKEv2/IPsec session as well as the time for reinstalling the context again in the same gateway.

Chapter 4 and 5 are dedicated to the Mono-LAN architecture. First, we analyze how ClusterIP (introduced in 1.3 and described in 2.5) improves the availability of a VPN service and then we show some results of our test-bed. In addition, we observe the drawbacks and benefits of this solution. Then, we define two architectures for IKEv2/IPsec environments: *High Availability* and *Traffic Management*. We present our implementations through GET and PUT functions, which are the heart of the IKEv2/IPsec context transfer.

In chapter 6, we address VPN platforms under Multi-LAN environments. We present a theoretical solution allowing to perform context transfer between different SGs owning different IP addresses. We show how the EU is impacted by the alteration of the IP address and how to solve this issue as well as other factors.

Finally, chapter 7 includes the conclusions of our investigations and the future work, as well as a description of the global evolution of the network concerning IPsec.

Part I

Security for IP Networks

Chapter 2

Roadmap of IPsec and IKEv2 protocols

This chapter provides an overview of the services, protocols and mechanisms that are involved in providing security based on IPsec and IKEv2. It also defines some mechanisms that provide mobility and flexibility for IPsec/IKEv2, as well as some methods that allow management and clustering at the IP layer. The goal of this theorical state of the art is to position most of the mechanisms that were used during practical tests and are often mentioned through next chapters.

2.1 Security Services

This section introduces some security services (also defined in [22]). Different protocols (e.g. TLS, IPsec, Https) might offer these services at different layers of the OSI model.

- *Data integrity*: the property that data have not been altered or destroyed in an unauthorized manner. In other words, during an IPsec-based communication, data integrity allows verifying that the contents of an IP datagram were not changed, either deliberately or due to random errors.
- *Data origin authentication*: the corroboration that the source of data received is as claimed. It allows a node to verify that each incoming IP datagram was originated by the claimed sender (source of data).
- *Peer entity authentication*: the corroboration that a peer entity in an association is the one claimed. This service involves the identification and authentication of the peer. Identification refers to an entity (user, equipment) claiming its identity by providing an identifier (usually a name, pseudonym, IP address, domain name). Authentication consists

in providing the claimed identity by providing one or several authentication elements, known as credentials.

- *Data confidentiality*: the property that information is not made available or disclosed to unauthorized individuals, entities or processes. As such, confidentiality transforms data from an intelligible format (plaintext) to an unintelligible format (ciphertext). It ensures that an unauthorized attacker or eavesdropper could not understand what is being transmitted during IPsec-based communications, this reverse action is known as decryption.

- *Replay detection*: consists for an entity to detect that received data are duplicated from a previous exchange. Some data might have been sent in a secure manner by a legitimate entity; but they can be copied and injected again to the same destination. Data are still authentic but they are already processed several times.

- *Access control*: the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner. For IPsec, access control concerns a host or a Security Gateway. Only authorized hosts or Security Gateways that are intended to establish IPsec communications will be allowed to do so.

2.2 IPsec: Security Associations and Databases

IPsec is a protocol that proposes to secure any IP based communication by providing security services (see 2.1). It is defined in [2]. IPsec offers interoperable, high quality and cryptographically-based security for both Internet Protocol versions: IPv4 and IPv6. The most common usage of IPsec consists in creating tunnels, often called Virtual Private Networks (VPNs). Actually, IPsec is used to create trusted environments through unprotected networks. When an entire network is being protected with IPsec, then *tunnel mode* is used. Otherwise, when the communication exclusively takes place between two hosts, then *transport mode* is more suitable. Figure 2.2 illustrates the scenarios where tunnel mode can be used, while figure 2.1 illustrates the transport mode scenario.

A Security Association (SA) is a data structure that contains security parameters to allow instantiation of an IPsec communication. The set of SAs is stored in the *Security Association Database* (SAD). Its goal is to offer security services for incoming/outgoing traffic it carries. As a SA is unidirectional, two SAs are needed in order to protect both ways of a bidirectional IPsec-based communication. In addition, some data are also stored in the *Security Policy Database* (SPD). The SPD is consulted to define appropriate behavior facing incoming and outgoing IP packets by specifying which traffic should be protected with IPsec or should bypass IPsec. It is consulted for all inbound and outbound traffic. The SA relies on Authentication

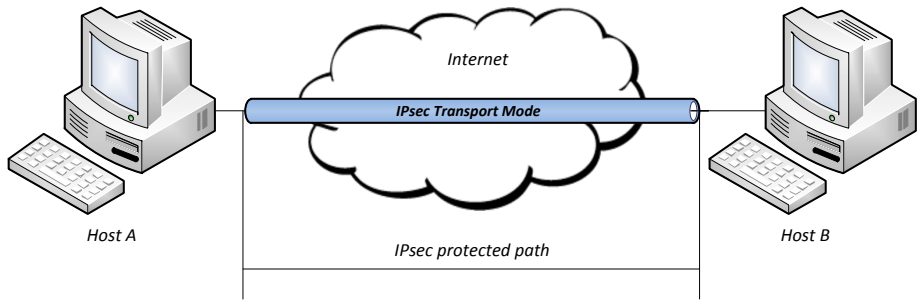
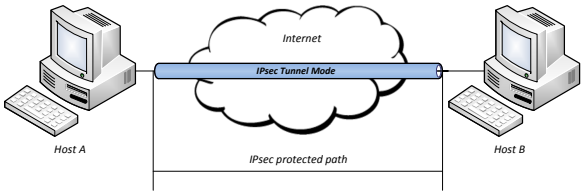
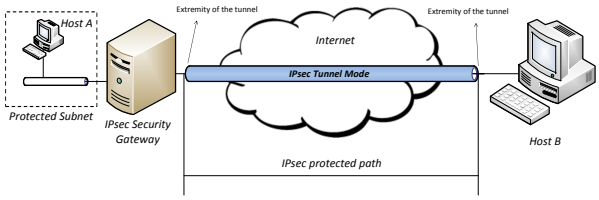


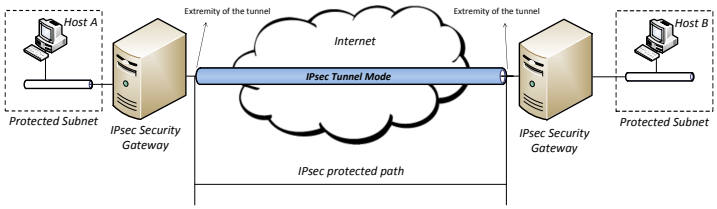
Figure 2.1: IPsec transport mode scenarios.



(a) Endpoint to Endpoint Tunnel



(b) Endpoint to Gateway Tunnel

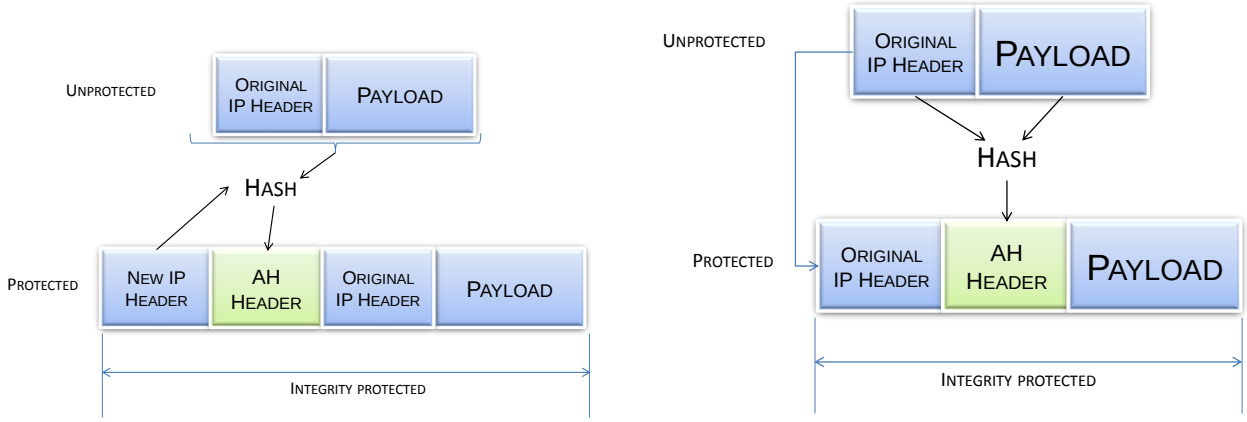


(c) Gateway to Gateway Tunnel

Figure 2.2: IPsec tunnel mode scenarios.

Header (AH; see section 2.3.1) and Encapsulating Security Payload (ESP; see section 2.3.2) protocols to offer its security services. Finally, all the parameters concerning SAD and SPD could be configured manually by an administrator or could also be configured dynamically thanks to the Internet Key Exchange Protocol (IKE). When IKE is used (which is mostly the case), an additional database known as *Peer Authorization Database* (PAD) links each authorized identity to its corresponding security parameters, thus, providing a link between IKE and each policy stored in the SPD. Further details concerning IKE are explained in 2.4.

IPsec can be deployed under different scenarios. Depending on the concerned topology, IPsec can be configured in two modes: *transport mode* or *tunnel mode*. Even if both modes use the same techniques and algorithms to encrypt/decrypt, the difference relies on the encapsulation and how a packet is built. Each mode has its own uses and thus it is important to care about the selected one in order to run IPsec properly.



(a) Authentication Header in Tunnel Mode

(b) Authentication Header in Transport Mode

Figure 2.3: IPsec Encapsulation Protocol: Athentication Header

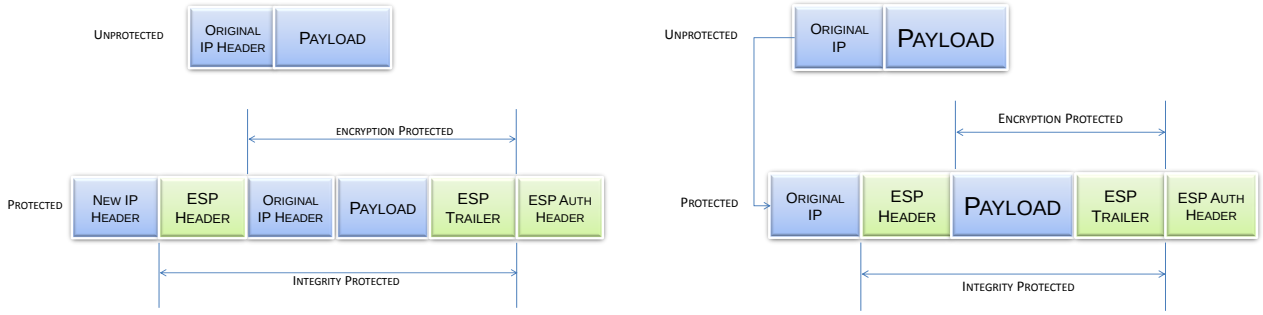
2.3 IPsec Protocols

2.3.1 Authentication Header

The Authentication Header (AH) is defined in [23]. It offers integrity, data origin authentication and an optional anti-replay protection. The AH header (as in fig 2.3), serves to transport the result of the integrity check over the entire original packet. As no encryption is performed, Authentication Header does not ensure confidentiality and consequently has a much simpler header than ESP. The AH header format depends on whether tunnel or transport mode is used. Figure 2.3 shows in detail how the AH header is built when the original IP packet is being protected using Authentication Header protocol in both transport and tunnel modes.

2.3.2 Encapsulation Security Payload

The Encapsulating Security Payload (ESP) protocol is defined in [24]. Depending on the configuration parameters, ESP can offer confidentiality, data origin authentication, data integrity and anti-replay services. The ESP header is more complex than the AH header. Figure 2.4 shows in detail how the ESP fields are built using both transport and tunnel modes. Notice that ESP not only adds a header between the IP header and the payload of the packet, but it also adds some fields at the end of the packet. So the fact that ESP fields enclose the original payload, makes security more complex than with AH which only adds an header.



(a) Encapsulation Security Payload in Tunnel Mode (b) Encapsulation Security Payload in Transport Mode

Figure 2.4: IPsec Encapsulation Protocol: Encapsulation Security Payload Header

2.4 Security Management: Manual Vs. IKEv2 protocol

Secrets are used as input keys for cryptographic algorithms. Naturally, the simplest idea concerning a shared secret would be to physically exchange it. Then, each peer can associate a host with its corresponding key. Unfortunately, this technique presents security risks when the exchange of secrecy is done under an uncontrolled environment. Additionaly, it is not scalable when deploying major infrastructures.

As mentioned in section 2.2, IPsec offers different security services by maintaining a shared state between the source and the destination of an IP datagram. This state defines the security services provided, as well as the cryptographic algorithms and their input keys. As cited above, maintaining this shared state in a manual fashion is not scalable at all. Hence, IKEv2 was designed to allow dynamic establishment and update of Security Associations, as well as the exchange of secret used as input to the cryptographic algorithms. IKE stands for Internet Key Exchange protocol. IKEv2 (defined in [25], [26]) is an improved version 2 of its predecessor IKEv1 (defined in [27], [28] and [29]). Throughout this thesis work, IKEv1 is not discussed at all because it is an obsolete version of the protocol. Henceforth, only IKEv2 is argued.

How does IKEv2 protocol work?

Every IKEv2 exchange consists in a pair of messages: a request and a response. A peer that sends the first request is called INITIATOR, while the other peer is called RESPONDER.

Initial exchanges of IKEv2 perform mutual authentication between the peers and establish an IKE Security Association (**IKE_SA**). An IKE_SA consists in a userland shared state used to negotiate further Security Associations for ESP and/or AH. Actually, these SAs for ESP and/or AH are referred as **CHILD_SA or IPsec_SA**. Figure 2.5 shows the IKEv2 exchanges in order to successfully establish an IPsec-based VPN.

The first exchange IKE_SA_INIT negotiates cryptographic algorithms, exchange nonces

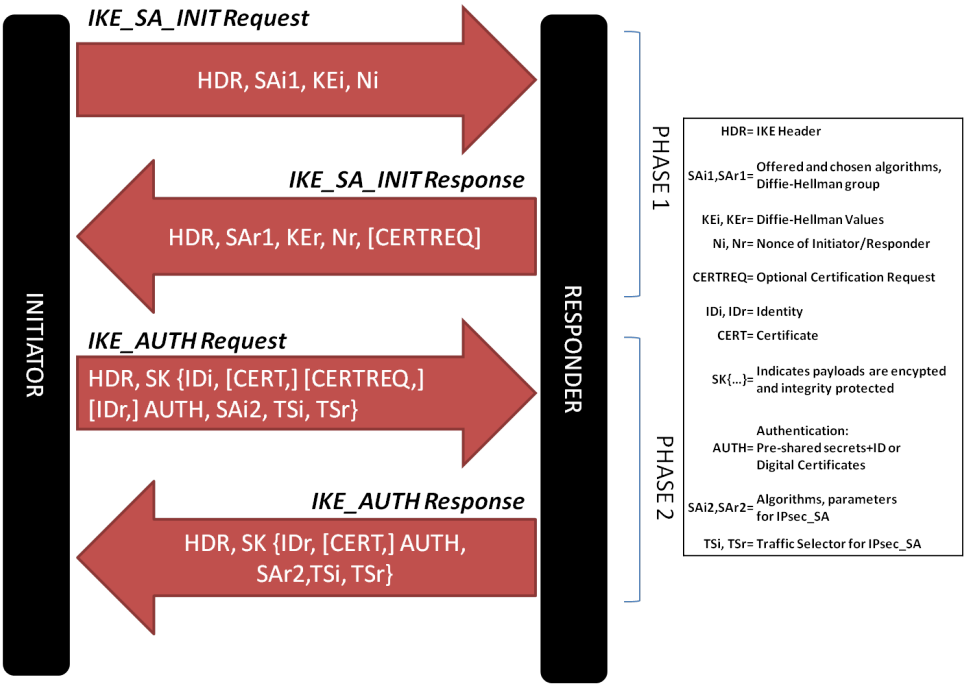


Figure 2.5: IKEv2 message exchanges, IKE_INIT and IKE_AUTH

and performs a Diffie-Hellman exchange. The following exchange IKE_AUTH transmits identities, demonstrates knowledge of the secret and negotiates the first CHILD_SA (i.e. the Security Association parameters). Only after both IKE_SA_INIT and IKE_AUTH have been acknowledged, further requests can be transmitted. The CREATE_CHILD_SA exchange allows to create a new CHILD_SA. And the INFORMATIONAL exchange allows to perform IKEv2 management tasks, like IP mobility, deletion of IKE_SAs, liveness verification, among others.

2.4.1 Exchange Description: IKE_SA_INIT

This exchange helps to establish trust settings which ensure future IKE exchanges. Figure2.6 shows an graphic explanation of the IKE_SA_INIT request and response.

The INITIATOR sends the following information:

- ***SPI***: security association unique identifier.
- ***Version Number***: set to version 2 (e.g. IKEv2).
- ***SAi1***: cryptographic algorithms proposed.
- ***KEi1***: Diffie-Hellman values.
- ***Ni***: INITIATOR's nonce.

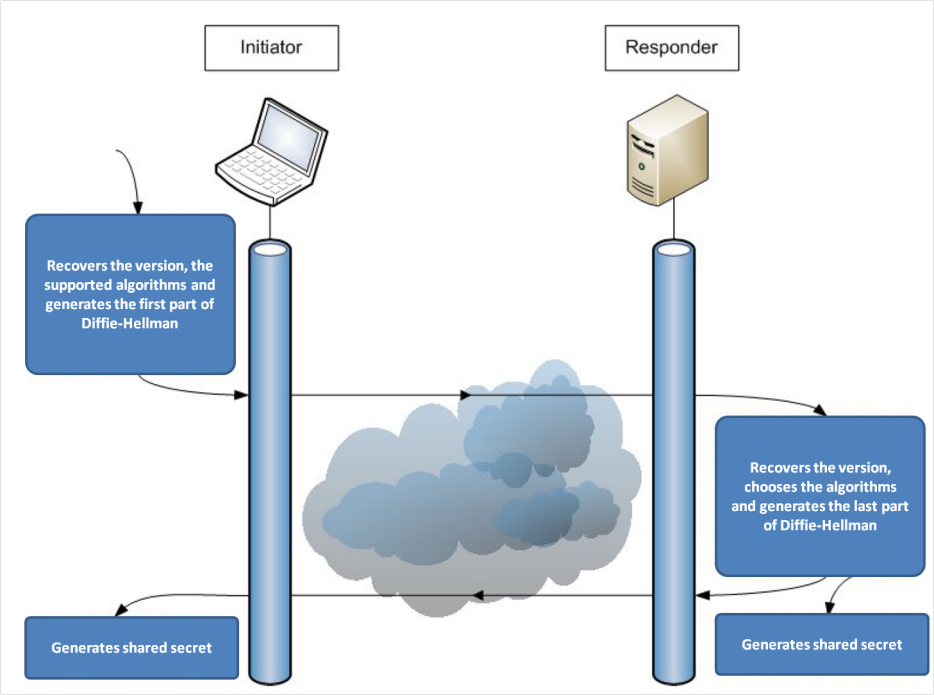


Figure 2.6: IKE_SA_INIT exchange details.

The responder replies with the following information:

- ***SPI***: security association unique identifier.
- ***Version Number***: set to version 2 (e.g. IKEv2).
- ***SAr1***: cryptographic algorithms chosen (based on proposition ***SAi1***).
- ***KEr1***: Diffie-Hellman exchange.
- ***Nr***: RESPONDER's nonce.
- ***CERTREQ***: if present, this optional field asks for a certificate.

At this point, both parties can generate a *SKEYSEED*, a shared secret from which all keys are derived for that IKE_SA. The first key, usually called SK_e, is used for encryption. The second one, for integrity, is called SK_a. And finally, the SK_d, is the key used to derive the keys for the CHILD_SAs (once again, keys for encryption and integrity protection).

2.4.2 Exchange Description: IKE_AUTH

During the IKE_AUTH exchange, both peers authenticate each other, and the first CHILD_SA is negotiated. Figure 2.7 illustrates the IKE_AUTH exchange.

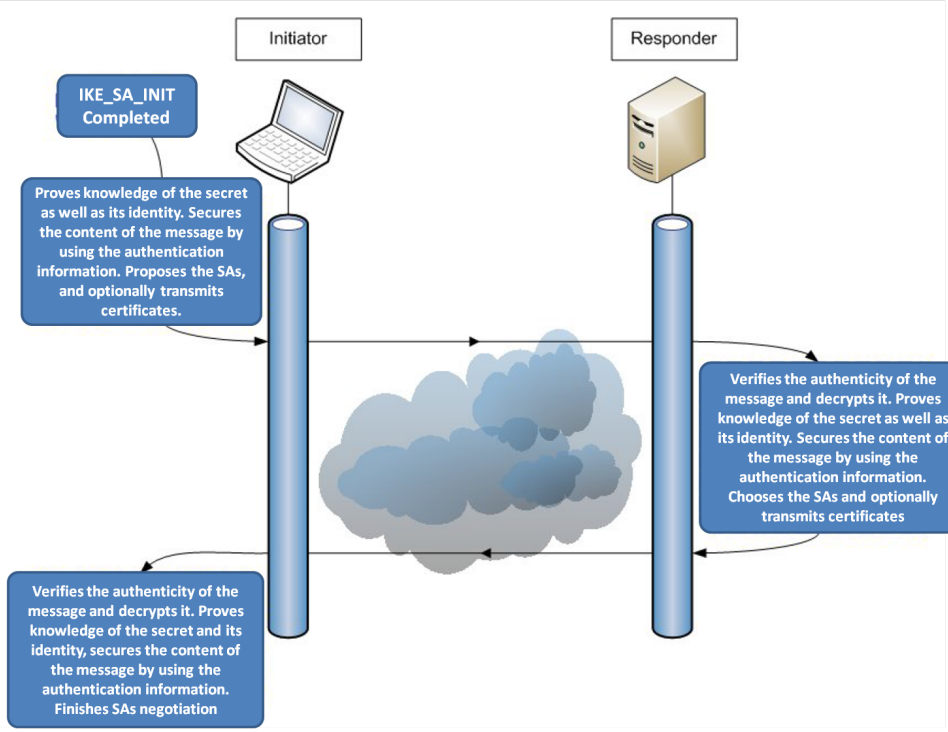


Figure 2.7: Detailed IKE_AUTH exchange.

The INITIATOR sends the request with the following information:

- ***SPI***: security association unique identifier.
- ***Version Number***: set to version 2 (e.g. IKEv2).
- ***IDi***: INITIATOR identity.
- ***CERT***: INITIATOR certificate(s) (i.e. public key).
- ***CERTREQ***: optionally ask for a certificate.
- ***IDr***: optional RESPONDER identity to specify which of the responder's identities it wishes to talk to (i.e. a host running multiple identities).
- ***AUTH***: a block of data protected with integrity protection (e.g. digital certificates or pre-shared keys + ID)
- ***SAi2***: cryptographic algorithms proposed for the CHILD_SA.
- ***TSi***: defines the source address of the traffic forwarded from the INITIATOR of the CHILD_SA.
- ***TSr***: defines the destination address of the traffic forwarded to the RESPONDER of the CHILD_SA.

The RESPONDER sends a response with the following information:

- ***SPI***: security association unique identifier.
- ***Version Number***: set to version 2 (e.g. IKEv2).
- ***IDr***: confirms identity by sending the RESPONDER's identity (IDr).
- ***CERT***: RESPONDER certificate with the public key to verify the AUTH field.
- ***AUTH***: all information which is used to perform integrity protection.
- ***SAr2***: cryptographic algorithms selected for the CHILD_SA.
- ***TSi***: defines the source address of the traffic forwarded from the INITIATOR of the CHILD_SA.
- ***TSr***: defines the destination address of the traffic forwarded to the RESPONDER of the CHILD_SA.

2.4.3 Authentication of IKE_SA

The authentication of the IKE_SA could be achieved in three different manners: Pre-shared keys, X.509 Certificates or EAP (Extensible Authentication Protocol).

The easiest method to configure and authenticate IKE_SAs is the pre-shared key, where both entities keep the same secret in a secure place. Security risks might be the main drawback of this authentication method due to the difficulty to keep the "shared" secret in a safe place. Also, the shared secret must be distributed among the entities that comunicate, which also represent a risk.

The authentication through certificate X.509 is another way to authenticate endpoints but demands a more complex system. The X.509 certificates use an algorithm based on assymetric cryptography. This method uses two cryptographic keys per user: a public key and a private key. The certificate authority certifies that both keys are unique per user and are properly linked. Depending on how these keys are used, the user can ensure integrity protection of the message as well as the identity of the sender. The sender signs the data using its private key and sends it to other users. Thanks to this signature, all recipients can verify the integrity and identity of the message by using the public key of the sender.

However, there are scenarios where the pre-shared key method does not scale well or the distribution of certificates X.509 is too complicated for the administrator (E.g. too much users). Thus, EAP-IKEv2 protocol consists in other way to perform authentication. EAP-IKEv2 uses an additional exchange towards an authenticator EAP server to authenticate endpoints. Details of EAP with IKEv2 can be found in [30].

2.5 Clustering at the IP layer: ClusterIP

ClusterIP is an implementation based on recent Linux kernels (2.6 and higher). It permits load-balancing features without having a physical load-balancer. It is a smooth way to distribute the load of incoming IP packets in a very autonomous manner (each node can decide whether it is responsible or not for each incoming IP packet).

However, let us first introduce the notion of a cluster. In a computer system, a cluster is a group of servers and resources that act like a single system. These are commonly used to offer high availability, parallel processing or load balancing. Cluster members may exchange some data in order to be able to offer these features.

Under Linux kernel distributions, *iptables* is an implementation which configures, maintains, and inspects firewalling rules. It identifies an IP packet that matches a defined rule. This action is called *target*. ClusterIP is an extended *target* module of *iptables*, allowing to build a cluster of machines that share a common IP address without having a physical machine that performs this task. Indeed, ClusterIP is intended to provide load-balancing and clustering features without having a dedicated load-balancer machine. Thus, ClusterIP consists in a loadbalancer-less cluster on Linux operative system. It simply enables a server to calculate responsibility of incoming IP packets so that the server can decide by itself either it has to treat it or not.

When using a ClusterIP approach, no special hardware is required to benefit from its load-balancing and clustering features. The configured members of the cluster share a common multicast MAC address and thus receive the same packets. Then, a lower-layer mechanism (ClusterIP) filters packets at the IP layer by calculating responsibility through an algorithm (E.g. hashing the IP source of each packet). When used in command line interface, ClusterIP is a parameter of the `iptables` command.

The following parameters are set when implementing ClusterIP:

- *–new*: creates a new cluster. Must be the first rule in the list of rules.
- *–hashmode mode*: mode of hashing. Must be one of the following: *sourceip*, *sourceip-sourceport*, *sourceip-sourceport-destport*.
- *–clustermac mac*: multicast MAC address of the cluster.
- *–total-nodes num*: total number of nodes within this cluster.
- *–local-node num*: the id number of the local node within the cluster.
- *–hash-init rnd*: a random seed for hash initialization.

The nodes in a cluster usually have two Network Interface Card (NIC). One of the NIC's MAC address is replaced by the shared cluster MAC address and then a common virtual IP address is mapped onto it. The other NIC, being completely independent, can be used for any other purpose, as for example inter-nodes communications (E.g. Hearbeat mechanism, keep alives, etc.). Given the case where a machine counts with only one NIC, it is also possible to install a second virtualized IP address on the same interface.

Originally, ClusterIP does not handle IPsec traffic. However, if an IKE daemon (E.g strongSwan) handles to synchronize IKE_SAs states and basic IPsec_SAs states, a modified version of ClusterIP that handles IPsec traffic could solve synchronization troubles. The overhead for synchronizing ESP sequence numbers is elevated due to the fact that ESP sequence counters grow too quickly. Deploying IKE and IPsec in a cluster requires the synchronization of some information among all the cluster members. Synchronizing counters might be the major barrier to overcome when it comes to setting up a cluster with IPsec.

2.6 MOBIKE - mobility extension for IKEv2

MOBIKE stands for IKEv2 Mobility and Multihoming (MOBIKE) Protocol. It is an extension to IKEv2 protocol. MOBIKE allows to change the IP address associated to an IKEv2/IPsec tunnel.

Originally, IKEv2 and IPsec states were designed more likely to be static. IPsec/IKEv2 data was supposed to remain the same during an IPsec session or VPN Tunnel establishment. For example, if a node that initially established an IPsec tunnel changes its original IP address (E.g. it change its point of attachment and changes its IP address), the node was forced to brake down its connection and to re-initialize a new IPsec tunnel from scratch with the new IP address.

MOBIKE allows to update the IPsec_SAs (SPD, SAD and PAD) and IKE_SAs (information contained in the IKEv2 application). Both peers need to agree on MOBIKE protocol support at the begging, which is is done during IKE_AUTH exchange of IKEv2 (see figure 2.8). When lower layers detect that a new IP address is present in the currently used interface, the peer updates its IKEv2/IPsec databases and it sends an INFORMATIONAL message including a notify payload UPDATE_SA_ADDRESSES. When the other peer receives this message, it updates its IPsec_SAs and IKE_SAs as well.

MOBIKE also offers some multihoming features. It allows to built, under priority order, a list of IP addresses where an endpoint is reachable. When connection is lost through the currently used IP address, the other endpoint can reach the peer through the second IP address on the list, then the third, and so on. Servers with more than one network interface can offer this feature. The notify payload which agreggates an IP address is ADDITIONAL_IP4_ADDRESS

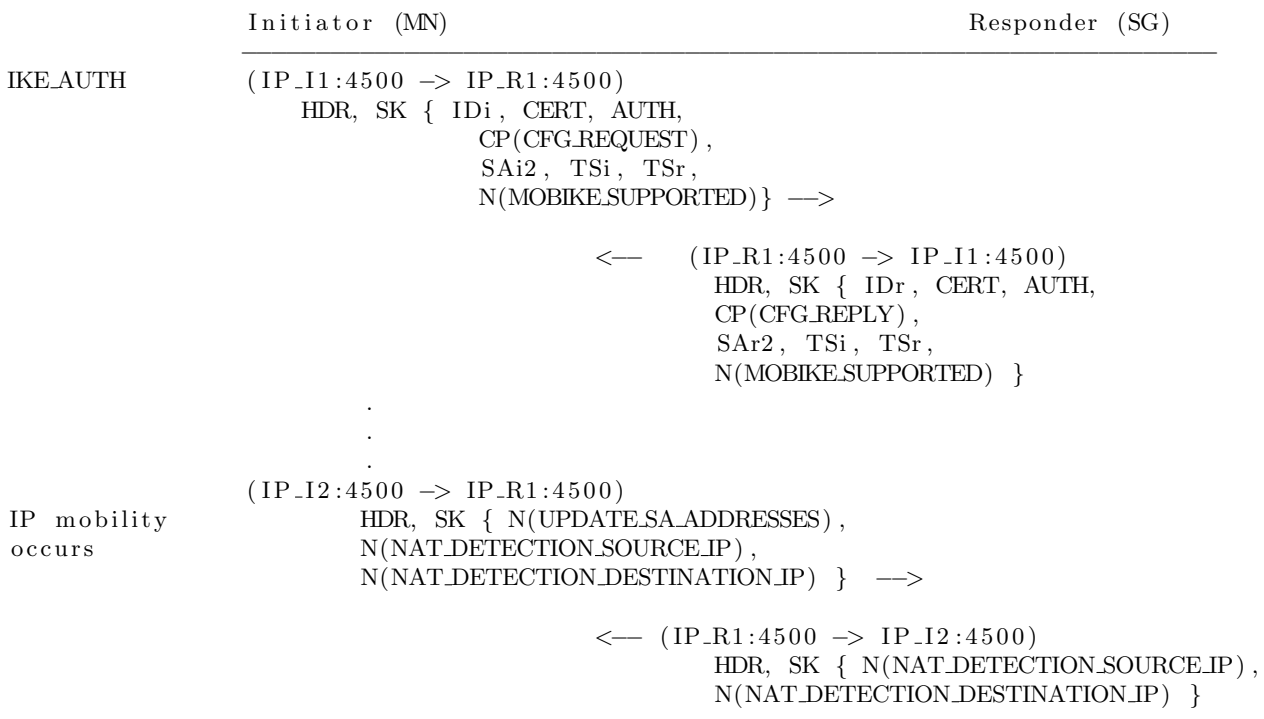


Figure 2.8: MOBIKE Protocol Exchanges

Figure 2.9: REDIRECT support during IKE_SA_INIT

or ADDITIONAL_IP6_ADDRESS whether it is an IPv4 or IPv6 address. If no additional IP address exists, then a notify payload NO_ADDITIONAL_ADDRESSES is sent.

2.7 REDIRECT extension for IKEv2

The REDIRECT mechanism is an extension of the Internet Key Exchange protocol (IKEv2) defined in [4]. It allows a Security Gateway to redirect the traffic flow to another Security Gateway (SG). Therefore, if a SG is being shut down for maintenance, it is possible to REDIRECT the VPN clients to another SG.

A peer can explicitly inform the other peers that it supports the REDIRECT mechanism by adding a notify payload during an IKE_SA_INIT or INFORMATIONAL exchange. Figure 2.9 shows the details during an exchange.

```
Initiator (MN)                                    Responder (SG)
-------------------------------------------------------------
                    <--  HDR, SK {N(REDIRECT, New_GW_ID)}

HDR, SK {} -->
```

Figure 2.10: REDIRECT exchanges

Figure 2.10 illustrates a REDIRECT exchange during an active session. The Mobile Node (MN) receives the INFORMATIONAL message with the correspondent notify payload and the new security gateway ID (i.e. IPv4, IPv6 or FQDN).

2.8 Conclusion

ISPs are mainly using IPsec Virtual Private Networks (VPNs) to extend a security domain over untrusted networks. End-Users may secure their communications with theirs ISPs by establishing an IPsec tunnel with a Security Gateway. Hence, ISPs are called to offer higly reliable services with IPsec Security Gateways. On the other hand, an operator might want to migrate an IPsec session from one SG to another for several reasons (e.g. offload, mobility, failure predictive actions, avoid re-authentication, cloud service topologies, among others).

Considering the evolution of mobile networks through the past ten years, ISPs need to face new challenges in terms of security. EUs are increasingly prone to mobility or multihoming operations. In addition, the new cloud services introduce more challenges in order to offer better QoS, as well as high availability to those contents that the EUs want to access wherever they need to.

Throughout this chapter, we introduce the notions of IPsec. Two entities wishing to establish a secure communication, may use the IPsec suite together with the IKEv2 protocol in order to authenticate and encrypt their exchanges.

Further explanations introduce some mechanisms that addresses new challenges facing mobility, multihoming and clustering at the IPsec layer. For example, we briefly introduce ClusterIP (see 2.5), REDIRECT (see 2.7) and MOBIKE (see 2.6) exentions, which improves the performances facing mobility, overload and multihoming issues of IKEv2/IPsec tunnels. Finally, these mechanisms are also extensively discussed later in chapter 6.

Chapter 3

IKEv2/IPsec Context Definition

3.1 Introduction

Operators are mainly using IPsec Virtual Private Networks (VPNs) to extend a security domain over untrusted networks. A VPN is usually established when an End-User (EU) and a Security Gateway (SG) negotiate some parameters referred as Security Associations (SA). For a better QoS, these SGs are geographically distributed so they are as close as possible to EU. As such, the higher is the level of responsibility of a SG, the higher is the risk to be overloaded and to break down.

This chapter introduces the definition of an IKEv2/IPsec context. When two nodes establish an IKEv2/IPsec-based communication or VPN, a set of parameters are agreed by both extremities of the VPN. In order to secure their IP traffic, some cryptographic information is built on both sides of the tunnel.

Initial exchanges first agree on a secure channel also called IKE security association (IKE_SA). This IKE channel permits IPsec entities to negotiate and maintain some security parameters known as IPsec Security Associations (IPsec_SA). An IPsec_SA consists in some shared information used to secure the IP communications between both extremities of a VPN.

Throughout this chapter, we will also present our investigations related to the extraction (commonly cited as GET) and re-installation (referred as PUT) of an IKEv2/IPsec context. Notice that both the IKE_SA and the IPsec_SA are referred together as IKEv2/IPsec context, which is simpler an easier for readers. On the other hand, all the developments of our implementation are done within strongSwan, which is an open source VPN software for Linux environments.

This chapter introduce a mechanism for extracting and reinstalling security associations (both IKE_SA and IPsec_SA), as well as a mechanism to transfer a given IPsec traffic from one SG

to another. We also propose an additional mechanism for solving the mis-synchronization of IPsec anti-replay counters and IKEv2 Messages ID counters. Finally some performance measurements are provided in terms of delays, and packet loss, and prove feasibility of the approach.

Section 3.3 describes details of an IKEv2/IPsec context. Section 3.4 introduces a proposed mechanism to perform IPsec Context Transfers (referred to as IPsec-CXT), a description of **GET** and **PUT** functions and a step-by-step description of an IPsec-CXT. Finally, sections 3.4.2 and 3.5 describe our implementation as well as the testbed results.

3.2 Motivation

IPsec provides a security framework at the network layer allowing IP and upper-layer protocols to benefit from encrypted and authenticated communications. They are mostly used to secure peer-to-gateway or gateway-to-gateway communications (see 2.1 and 2.2), or VPN (Virtual Private Networks). When providing VPN services based on IPsec, an End-User (EU) commonly makes use of the IKEv2 protocol [26] to negotiate IKEv2 security associations (IKE_SAs) and further IPsec security associations (IPsec_SAs) towards a security gateway (SG). Figure 2.5 illustrates the tunnel establishment and different exchanges.

In order to reduce delays and improve communications, the path between an EU and the SG should be as short as possible. As such, operators may rely on geographically distributed pool of SGs to provide a high QoS. On the other hand, an operator that offers fail-over capacities should be able to transfer VPN sessions from one SG to another. Motivation may be to spread the workload among different SGs, as well as to provide high availability features in case a SG fails. The scenario we are considering is an EU that sets up a VPN towards an SG (e.g. SG1) which gives access to a trusted network. The EU is first authenticated by using IKEv2 protocol with either pre-shared keys, EAP-AKA, EAP-SIM, certificates, etc. Then, further IKEv2 exchanges allow to set up the VPN (i.e. IKE_SAs and IPsec_SAs). At a given time, SG1 is overloaded, whereas some other SGs are not (e.g. SG2). The operator wants to move this VPN session from SG1 towards SG2, so it transfers all the IKEv2/IPsec context. The main goal of transferring all IKEv2/IPsec context is to avoid re-authentication against SG2. In fact, re-authentication would increase signaling and would interrupt the communication which could be critical for real-time based applications. Note that if SG1 and SG2 do not have the same IP address, then the EU MUST also update its IKE_SAs and IPsec_SAs. Transferring the IKEv2/IPsec context introduces the following constraints: (1) identification of the IKEv2/IPsec parameters so it can be extracted and reinstalled into another SG. Notice that some parameters of the context requires perfect timing, which is much more difficult to get than the IKEv2/IPsec context itself, (2) two gateways with different IP addresses bring complexity as the hosts should consider the management of their tunnel and their SAs (mobility aspect), as well as the distance between SGs introduces

network delays.

3.3 IKEv2/IPsec Context Detailed Description

The IKEv2/IPsec context contains all the information negotiated through IKEv2 protocol as well as all the information contained in the IPsec databases. Most of the current operating systems implement the SAD and SPD in the kernel while the PAD is mostly a database that exists in the userland space. The SAD, SPD and PAD are defined as follows:

- **Security Association Database SAD**: contains all the information related to each unidirectional IPsec_SA (SPI, concerned protocols, SA's lifetime, algorithms, keys, etc.).
- **Security Policy Database SPD**: defines how the packets that match the IPsec policies are handled by the device: PROTECTed, BYPASS or DISCARD IPsec.
- **Peer Authorization Database**: links the SPD with the key management protocol (i.e. IKEv2). This database determines if an identity is allowed or not to establish a VPN with the device.

The Security Associations (SAs) contained in the SAD can be configured manually, but this is obviously not scalable when big amounts of IPsec connections are established with different devices. Thus, IKEv2 (see [26]) is the protocol that sets a secure channel (a.k.a. IKE_SA) between two end-points in order to negotiate and continuously update all the IPsec related information (a.k.a. CHILD_SA or IPsec_SA) between them. First, during IKEv2 initial exchanges (known as *Phase 1*), one of the end-points (INITIATOR) triggers the communication by sending an IKE_INIT request. The other end-point (RESPONDER) replies back with an IKE_INIT response. At this point, both nodes have a shared secret to perform encryption and integrity protection for further IKEv2 exchanges, and have agreed on the following parameters of their IKE_SA:

- *Cryptographic algorithms*: algorithms to protect further IKE exchanges, a Diffie-Hellman Group and a pseudo-random function.
- *SKEYSEED*: secret key from which all keys are derived for that IKE_SA (i.e. SKe encryption key to ensure confidentiality, SKa authentication key to ensure integrity and SKd derivation key master secret that will be used to compute further CHILD_SAs keys).
- *IKE_SPI*: IKE_SPI stands for IKE Security Parameter Index. It uniquely identifies an IKE_SA.
- *Lifetime*: duration of an IKE_SAs.

- *Nonces*: the INITIATOR nonce and RESPONDER nonce (Ni and Nr) are randomly generated values which reinforce the security by refreshing the key material.
- *Message ID counters*: the ID counters provide anti-replay for IKEv2 exchanges by increasing the ID counter by one for every emitted IKEv2 message.
- *IKEv2 window size*: if the window size has a value of "N", it implies that there can be N un-acknowledged IKEv2 requests at any given time during communication.

During the *Phase 2* of IKEv2, the INITIATOR sends a IKE_AUTH request and the RESPONDER replies with an IKE_AUTH response. Now, the nodes consult their PAD in order to authenticate each other. As mentioned, the PAD determines if the identity of a given node is allowed or not to establish a CHILD_SA. The PAD is composed of the following information:

- *Identifiers*: IDi (INITIATOR ID) and IDr (RESPONDER ID). Usually an IPv4/IPv6 address, a fully-qualified domain name, an email address,etc.
- *Authentication Method*: pre-shared key, EAP, certificates, RSA, etc.

The establishment of the CHILD_SA (negotiation of parameters stored in the SAD and SPD) is piggybacked during the IKE_AUTH exchange. It is done just once the authentication and the authorization are performed. When *Phase 2* is done, both nodes agree on the following parameters of their CHILD_SAs:

Concerning the SAD

- *CHILD_SA SPI*: a 32 bits unique identifier of the CHILD_SA.
- *IP addresses*: source/destination IP addresses of the IKEv2 compliant nodes.
- *IPsec Protocol*: AH (Authentication Header) or ESP (Encapsulating Security Payload).
- *Sequence number counter*: value to control every incoming/outgoing IP packet protected with IPsec, preventing replay or unauthorized re-injection of already processed IPsec traffic.
- *Anti-replay window size N*: any packet with the sequence number $X + N$ is discarded, where X is the awaited sequence number.
- ESP/AH information: Encryption and/or authentication algorithms, keys, initialization values, key lifetimes.
- Lifetime: time interval or byte count after which a SA must be replaced with a new SA (and new SPI).

- IPsec Protocol Mode: tunnel or transport mode.
- Path MTU: maximum size of an IPsec packet that can be transmitted without fragmentation.

Concerning the SPD

- *IPsec Protocol Mode*: tunnel or transport mode.
- *Header's IP Address*: source/destination IP addresses of the IKEv2 compliant nodes.
- *Source/Destination IP addresses of the communications*: if transport mode is being used, these addresses are the same as those used for routing purposes, whereas in tunnel mode these IPs concern the end-points of the communications (which could be an internal IP address of a protected subnet).
- *Upperspec*: upper-layer protected protocol (HTTP, FTP, HTTPS, among others).
- *Policy rules*: DENY, BYPASS or PROTECT the targeted traffic.

3.4 GET and PUT functions: Description

3.4.1 Motivation: manage and transfer of and IKEv2/IPsec context

We propose a framework (fig. 3.1) to dynamically manage an IKEv2/IPsec context. New features are introduced in section 3.4.2 (GET and PUT). This framework proposes to extract and re-install an IKEv2/IPsec context from one SG to another, which we refer as IPsec-CXT (*IPsec context transfer*). The IPsec-CXT between two SGs can be performed with CXTP protocol (see [7]), mobility operations (e.g. MOBIKE) or with REDIRECT extension of IKEv2. However we will first concentrate in just extracting and re-installing the IKEv2/IPsec context before its transfer. The transfer of an IKEv2/IPsec context is discussed in detail in chapter 4 and 6.

In order to improve the management of an IKEv2/IPsec , we initially define two operations: **GET** and **PUT**. **GET** extracts the IKEv2/IPsec context for a given tunnel, whereas **PUT** makes installation of it. Details of an IKEv2/IPsec context are described in [18].

3.4.2 Implementation

Our implementation is based on *StrongSwan 4.5.0* [16], a widely used IKEv2 open-source implementation for linux environments. Its multi-threading design meets today's requirements. As IPsec is a complex protocol, *strongSwan* has a very modular style. A daemon called **starter** initializes the IPsec framework and launches the different daemons defined through configuration

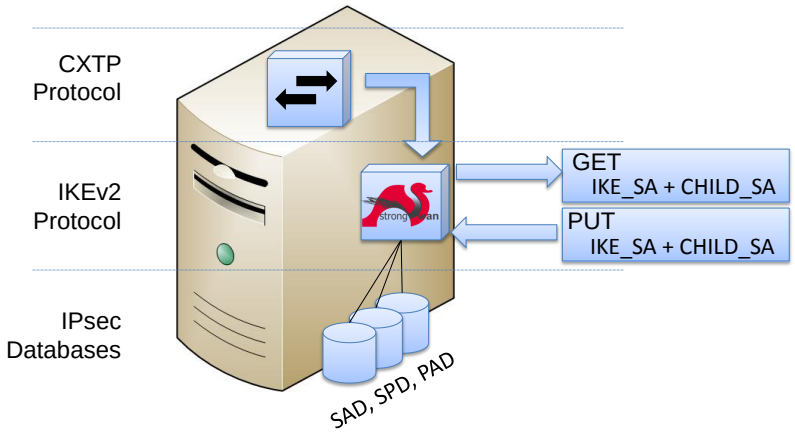


Figure 3.1: Schema of an IKEv2/IPsec Context Transfer Mechanism in a Gateway

files (i.e. *ipsec.conf* where most of the parameters are stored). The daemon responsible of IKEv2 is called **charon**. It intercepts the IP packets at the IP stack and applies the security policies (whether to perform encryption/authentication or not). This daemon is initiated as a thread, a master daemon creates simultaneous processes and feeds them with different information. A data bus is created in order to interact with the user interface. This data bus is called **stroke**, and it interacts with **charon**. Thus, this is how *strongSwan* works as a highly parallelized application.

StrongSwan is configured through the following files:

- **strongswan.conf** defines the parameters of *strongSwan* at launching time.
- **ipsec.conf** stores the configuration concerning each CHILD_SA (i.e. how the connection must be secured).
- **ipsec.secrets** defines the PAD which contains the parameters required to authenticate the peers. It also contains information related to the private keys and the methods to perform authentication.

StrongSwan can be launched through command-line interface with root privileges as follows: `ipsec start`. Once *strongSwan* is launched, a tunnel is initiated by running the command `ipsec up <conn-name>` (`<conn-name>` is the name of the session as in the configuration files).

We implemented two functions in order to successfully recover a whole IKEv2/IPsec context. That is, we **GET** a context and store it into a plain text file. We also implemented a function to **PUT** (re-install) an IKEv2/IPsec context into a SG directly from the previously stored plain text file (see figure 3.1 to observe the graphical representation). Both **GET** and **PUT** functions are implemented in *strongSwan*, which communicates with the kernel through the Netlink XFRM API in linux. This API has recently been modified in order to solve issues

concerning the IPsec replay sequence counters synchronization. Netlink XFRM allows to modify these counters at any time in order to synchronize them. This feature is very useful when clustering security gateways. On the other hand, the lookup of the IKE_SA and CHILD_SA inside *strongSwan* is done through its connection name (we refer to it as "`<conn-name>`" in the examples): the command `ipsec get <conn-name>` performs **GET** whereas the command `ipsec put <conn-name>` performs **PUT**.

- `ipsec get <conn-name>`: The command `ipsec` is intercepted by the *strongSwan* daemon. The command `get` launches the function that recovers the IKE_SA + PAD + CHILD_SA. The lookup of the IKE_SA and its corresponding CHILD_SA is done through its name "`<conn-name>`". Finally, it creates a plain text file in the temporary directory (`/tmp/`) with all the information.

- `ipsec put <conn-name>`: The command `ipsec` is intercepted by the *strongSwan* daemon. The command `put` launches the function that installs the IKEv2/IPsec recovered context from a plain text file. The IKE_SA and CHILD_SA are introduced in *strongSwan* with the name:"`<conn-name>`".

3.4.3 Testbed

Our platform is composed of two desktop stations. Both operating systems run over Ubuntu (v12.04LTS in the IPsec EU, and v11.10 in the SG). Our tests are performed locally on the same SG and permit to test the interaction with the IPsec stack.

First of all, we focused on measuring the time to establish a single VPN session towards a SG. Then, we increment the number of VPN session to perform load tests on the platform. We also measure how an interruption of service at the IPsec layer impacts upper-layers transmissions (like HTTP). For example, refer to figure 3.5a to see in details all the exchanges while performing **GET** and **PUT** at the IPsec layer during and HTTP download. We focus in studying the impact of this interruption. Tests are performed with HTTP traffic over Ethernet links.

The results are represented in graphs with a box-and-whiskers style (also called quartiles). This kind of representation is mostly used to plot statistical data. For every measurement made (around 100 samples per measure), the box-and-whisker figure indicates: (i) the smallest observation, (ii) the lower quartile, (iii) the upper quartile, (iv) the largest observation and (v) the median. The space between the lower and upper quartile represents 50% of the samples (see figure 3.2).
During the tests, we used Traffic Control in order to vary the bandwidth limit during the HTTP download and thus study the impact over different transmission rates. To generate statistical results, 100 measurements were done.

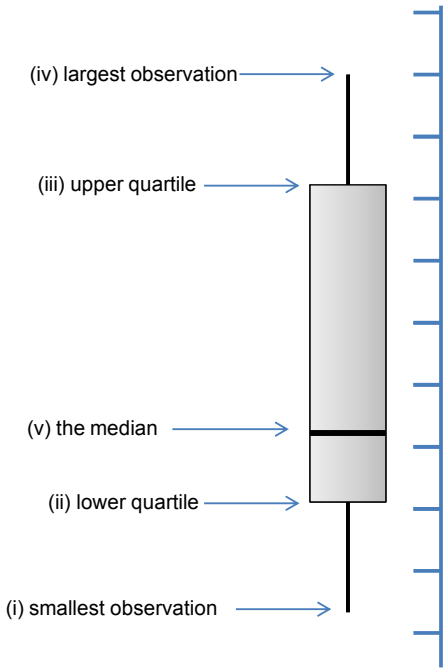


Figure 3.2: Box-and-Whisker plot representation (quartiles)

The tests have been measured by using the command `time`, which writes a message to standard output giving timing statistics about this program run. The outputs of `time` are (i) the *elapsed real time* between invocation and termination of the command, (ii) the *user CPU time* and (iii) the *system CPU time*. We considered the value of the *elapsed real time* in order to evaluate our stats results. For simplicity, we will refer to *elapsed real time* as `time`. This is how we measured the time it takes for a given command to run.

3.5 Performance tests & Results

The different testbed measurements are performed over Ethernet links (some of them with different bandwidths). We load the SG by establishing different numbers of VPN tunnels for each TCP connection granulated by its port number. For example, a TCP connection using port 80 will not traverse the same tunnel as a TCP connection using port 81. They will use different cryptographic material and thus different IKEv2/IPsec tunnels.

3.5.1 First Test - VPN Establishment Time

The first test measures the time for establishing IKEv2/IPsec tunnels between two end-points (see figure 3.3). The tunnels were initiated one after the other to avoid half-opened IKE-SAs.

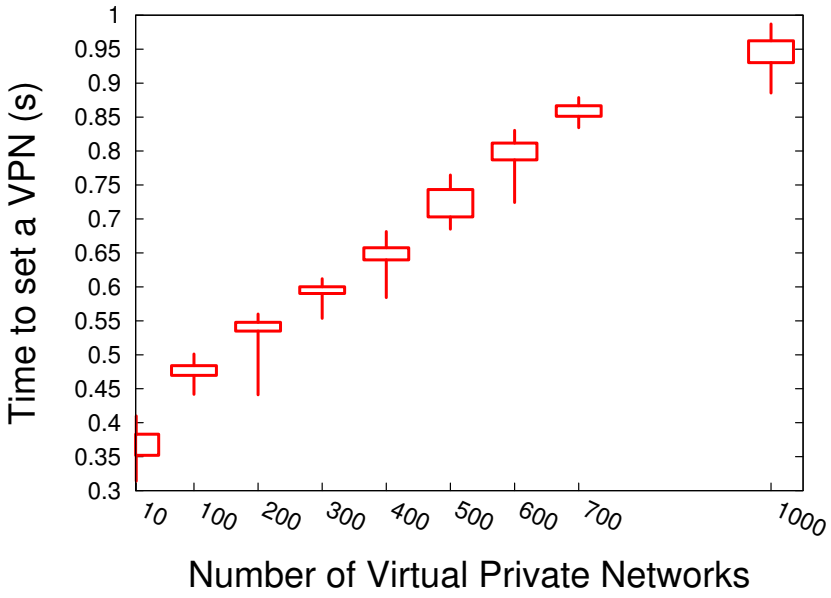
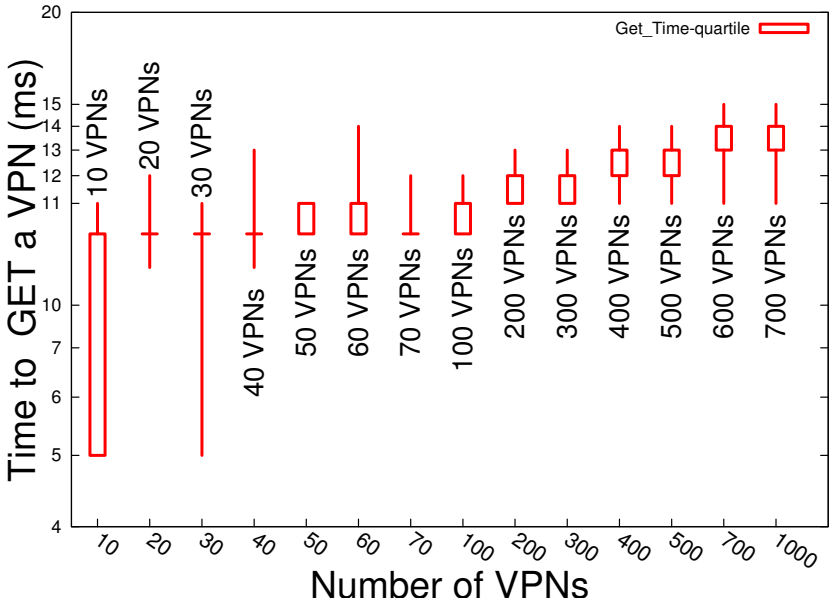


Figure 3.3: VPN Establishment Time

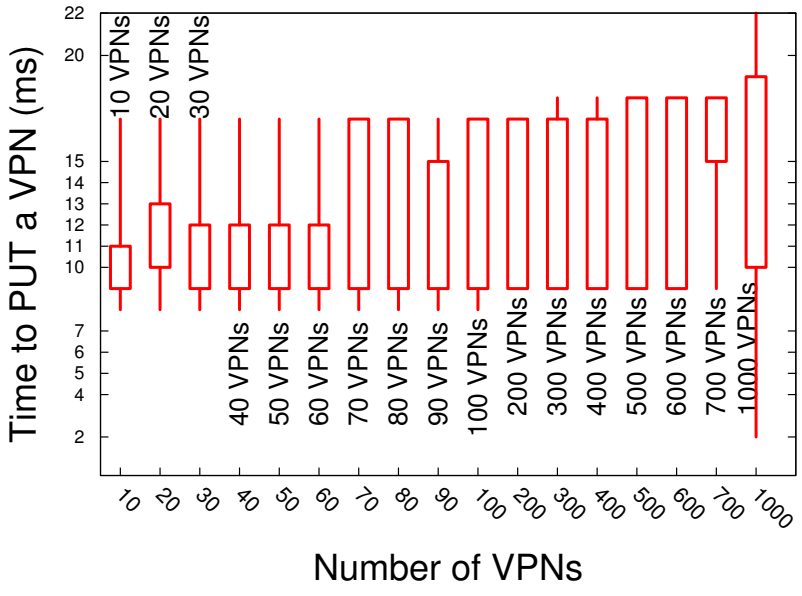
This is not a stress test but a load test. Figure 3.3 gives the results for initializing 10, 100, 200, 300, 400, 500, 700 and 1k tunnels. We can clearly see that the time measured is proportionally higher as the amount of VPNs rises. Establishment of 10 tunnels takes 0.31s to 0.41s, whereas 1000 tunnels may take 0.89s up to 1s for a single VPN establishment. All the other measures (i.e. 100, 200, 300 tunnels ...) show results between 0.3s and 1s for a single VPN establishment.

3.5.2 Second Test - GET and PUT Times

Our second test consists in measuring the time to **GET** and to **PUT** an active IKEv2/IPsec context. First, we established different scenarios (different amount of active tunnels): 10, 20, 30, 40, 50, 60, 70 , 100, 200, 300, 400, 500, 600, 700 and 1k active VPN tunnels. Then, we proceed to **GET** ten (10) randomly chosen VPNs over each scenario (10,100,200...). The resulting time to **GET** 10 randomly chosen VPNs is divided by 10 in order to obtain the average time to perform a **GET** over a single VPN connection. Detailed results are shown in figure 3.4a. We can observe that the values are quite similar for all the scenarios (we consider the case of 10 and 30 VPNs isolated variations, which is normal). The command `ipsec get` performs its task in a range of $5ms$ up to $15ms$. By the way, by observing figure 3.4a we can realize that a loaded SG (600-700 VPNs) takes more likely $13ms$-$15ms$ to perform a **GET** over a single VPN whereas for a not loaded SG these times are around $5ms$ up to $12ms$. This dispersion can be explained due to the semaphores for controlling access to an IKE_SA. This means that an IKE_SA can be in use by any process within strongSwan, and thus increasing the time to GET the desired IKE_SA.

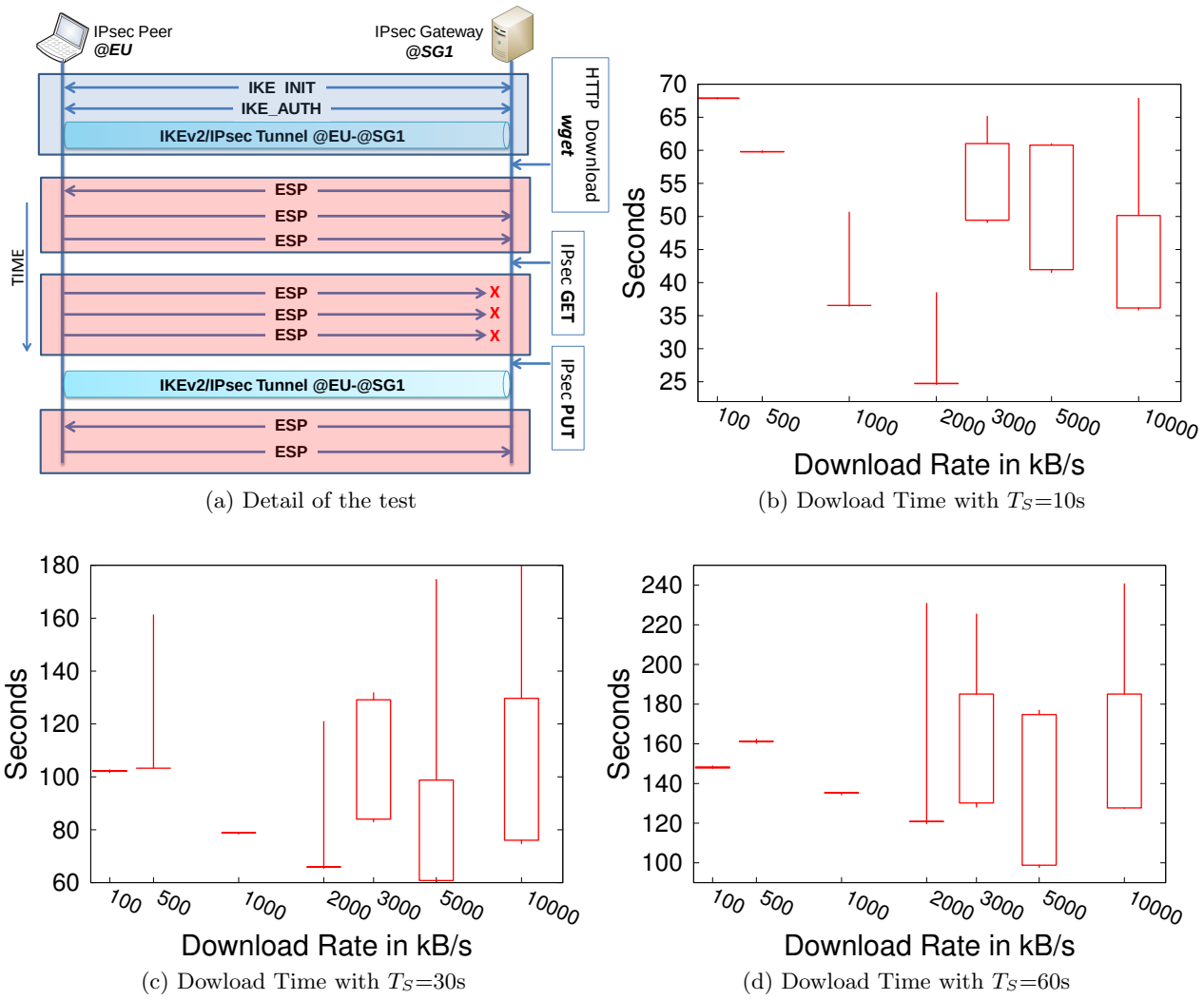


(a) **GET** Test



(b) **PUT** Test

Figure 3.4: Experimental IKEv2/IPsec Context Management



(a) Detail of the test

(b) Dowload Time with T_S=10s

(c) Dowload Time with T_S=30s

(d) Dowload Time with T_S=60s

Figure 3.5: Upper-layer's Reactivity

Then, we proceed to **PUT** these ten (10) previously randomly chosen VPNs for each scenario (10,100,200...). The time that the script takes to **PUT** these connections is divided by ten (10). This results in the average time to **PUT** a single VPN active session. Figure 3.4b shows the measurements concerning the **PUT** function for all the scenarios. It represents the time that a SG takes to install a single IKEv2/IPsec context from a file. We observe that *strongSwan* takes a range of $2ms$ up to $22ms$ to perform a **PUT** of a single VPN.

3.5.3 Third Test - Upper-layer's Reactivity

We observed the reactivity of upper-layer protocols facing interruptions at the IPsec layer on SG's side. The test consists to **GET** and **PUT** a single VPN connection during an HTTP download. We perform this with different traffic rates (controlled with Traffic Control implementation for

linux [31]). Figure 3.5a illustrates a protocol representation of what happens during this test. Initially, a VPN is established between an EU and the SG. Then, we start an HTTP download by using the command `wget`. The source file is placed on the EU's side whereas the SG is configured as the destination. Even if in real life, an EU (and not the SG) is usually the destination when downloading files from the Internet, we chose this methodology in order to analyze the impact of a SG facing interruptions at the IPsec layer. Thus, after the download has been performed during five (5) seconds, we **GET** the whole IKEv2/IPsec context on the SG, causing it to loose connectivity with the peer. The EU continues to send ESP packets as it is unaware of the **GET** function performed on the SG side. We considered different interruption times (T_S Time_Sleeps of 10, 30 and 60 seconds) before performing a **PUT** function. Finally, once the IKEv2/IPsec context is reinstalled, we observed the time to finish the HTTP download. As we carry HTTP traffic over an IPsec protected communication, we set the HTTP parameters through `wget` to be as flexible as possible facing interruptions. Also, as we measured different traffic rates during the downloads, we changed the size of the file being downloaded (because big size files would take too much time to download at lower rates). Table 3.1 shows the different file sizes used for each traffic rate. Figure 3.5 represents the download time concerning all file sizes and rates. Each

Table 3.1: Third Test - Interruption Times, Traffic Rate and File Sizes

Interruption Time T_S	Rate	File Size
10s, 30s and 60s	10 kB/s	1MB
	100 kB/s	5MB
	500 kB/s	20MB
	1 MB/s	20MB
	2 MB/s	20MB
	3 MB/s	100MB
	5 MB/s	100MB
	10 MB/s	100MB

figure (3.5b, 3.5c and 3.5d) shows the download times for each interruption T_S. We observe that at lower traffic rates (less than 3000kB/s), the measures are very stable because the quartiles are very thin. For higher traffic rates (more than 3000kB/s), the download time is unstable and so the quartiles are more spread and thicker. On the other hand, the three graphs have a similar behavior. Even though they are all offset by their corresponding interruption time. Finally, this test represents the impact of the interruption at the IPsec layer during an active session.

3.6 How can we transfer an IKEv2/IPsec context?

There are different approaches for transferring an IKEv2/IPsec context:

- **RFC4067 Context Transfer Protocol (CXTP)** [7] introduces a generic mechanism that allows context transfer between SGs. In our scenario, Context Transfer Protocol (CXTP) is the protocol we could use to transfer an IKEv2/IPsec context between Security Gateways whatever the context is. One issue when using CXTP under this scenario is that the EU is actually involved during the SG migration, thus increasing the number of messages exchanged between the EU and the IPsec SG. CXTP can be triggered by one of the SGs (network controlled) or by the EU itself (mobile controlled, or controlled by the EU). In both cases, a message named CTAR (Context Transfer Activate Request) must be sent from the EU towards one of the IPsec SGs during the context transfer.

- **Mobility Related Documents**: [3] and [32] address EU's IP mobility and multihoming (this is clarified in 1.3). When an EU changes its IP address, the IKEv2/IPsec contexts are no longer valid and the EU looses connectivity. An IKEv2 INFORMATIONAL exchange allows all concerned SAs to be updated and thus let the node remain connected and protected with IPsec. This might be confusing with IPsec-CTX except that IPsec mobility concerns the same SG, whereas IPsec-CTX concerns different SGs. However, mixing IPsec-CTX with mobility operations can be an approach to transfer an IKEv2/IPsec context. We will consider this in chapters 6.

- **Allard** [11] addresses the IPsec context transfer between two Security Gateways with an implementation using IKEv1 and CXTP protocol. In contrast, our work is positioned using IKEv2, which involves differences compared with IKEv1 (see section 2.3.1 in [33]): less signaling, mobile friendly ([3] and [32]), better performance and less complexity.

- **RFC5685 Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)** [4] proposes an extension to IKEv2 which redirects an IKEv2/IPsec session from one gateway to another. It does NOT pre-authenticate the EU prior to the connection towards a new Security Gateway (SG). The EU needs to renegotiate a new SA with the new SG.

3.7 Conclusion

We introduced the IKEv2/IPsec context and we proposed a mechanism to dynamically GET and PUT an IKEv2/IPsec context using strongSwan. We studied through a real implementation the time to establish an IKEv2/IPsec session, as well as we measured the time to extract and re-install a security context. Finally, we measured the impact for upper-layers through different interruption times at the IPsec layer during an HTTP-based download. We show the impact of GET and PUT fucntions locally. Following chapters will consider measuring the impact of the transfer of an IKEv2/IPsec context between physically different SGs. Additionally, for the use

cases where different SGs own different IP addresses, other extensions of IKEv2 (i.e. REDIRECT or mobility operations) are referred. Additionally to **GET** and **PUT** operation delays, we will consider the added network delays when performing an IPsec-CXT.

Regarding a classic network behavior, we consider the results of **GET** and **PUT** functions as optimistic. Based on the results in section 3.5.2, the worst case to perform a **GET** and a **PUT** are $15ms$ and $22ms$ respectively. This means that we can reestablish a tunnel within $37ms$ ($15ms$ to GET + $22ms$ to PUT). However, in chapters 4 and 6, we will consider the time while transferring the IKEv2/IPsec context from one SG to other. Finally, for those scenarios where the SGs have different IP addresses, we need to consider the time to update the Security Associations at the EU's side too.

Part II

IPsec/IKEv2 Context Transfer for Mono-LAN architectures

Chapter 4

Mono-LAN Context Transfer: ClusterIP

This chapter introduces the IKEv2/IPsec context transfer between different SGs with a shared common IP address. This is what we call a **Mono-LAN** configuration. We concentrate on studying and configuring a VPN platform (using strongSwan) based on a mechanism called ClusterIP, which offers High Availability together with load-balance features. The packets going towards the cluster are spread among the cluster's members according to a hash of the End-User's IP address. We demonstrate that ClusterIP improves the availability of upper-layers services, even though it is limited to the implementation within a same network segment, as it is required to maintain a shared common IP address based on Multicast MAC addresses.

We concentrate on how to build a cluster of SGs based on a HA plugin designed by strongSwan, which uses a target extension of *iptables* for Linux-based systems (ClusterIP). We describe the main issues to overcome HA within IPsec. We also measure how HA events may affect the EU experience, and provide recommendations on how to deploy ClusterIP. During an active IKEv2/IPsec tunnel, we simulate a failure on the active SG and evaluate the impact over upper-layers while the ClusterIP mechanism automatically switches from the active SG to the passive SG.

Note that the IP address remains the same while transferring an IKEv2/IPsec context in a Mono-LAN environment, these actions are transparent to End-Users. They are actually interpreted as a simple interruption of the communication and no signaling is needed between the End-User and the SG.

Section 4.1 presents the motivations and introduce the Mono-LAN scenario. Section 4.7.1 define our testbed. Section 4.3 presents the related work. Then, section 4.4 describes the challenge to overcome VPN counters synchronization during an IPsec context transfer. The following section 4.5, explains how to set up the platform with ClusterIP. Section 4.6 explains how

strongSwan implements ClusterIP in order to create IPsec SG clusters. Section 4.7 shows the experiments performed as well as the results obtained. And section 4.8 gives our conclusions.

4.1 Motivations and Mono-LAN High-Availability Scenario

Nowadays, mobile operators are facing the exponential growth of mobile data. Among different solutions, a short term alternative consists in switching mobile data from Radio Access Networks (RAN) to WLAN network. For example, the iWLAN [34] architecture proposes to carry the IP data between the End-User (EU) and the operator's core network through a WiFi access. Once the EU is transferred to the WiFi access, it establishes an IPsec connection with a dedicated Security Gateway (SG) that gives access to some services or simply to the Internet. Furthermore, this IPsec session should maintain the QoS prior to offload and the SG must remain available and reliable to EUs.

The industry is increasingly interested in providing services with High Availability (HA) capacities. The HA clusters aim to increase the availability of a service. In terms of IPsec, an HA IPsec cluster increases the IPsec service's availability. For example, when an IPsec tunnel is established towards a given SG within a cluster, the IPsec parameters might be spread among its cluster members. In the event of a failure during an IPsec session, some other SGs must ensure the VPN service. Actually, the availability and the continuity of service are guaranteed by different mechanisms.

It is important to distinguish a mobility event from an IPsec context transfer due to a failover. We have already discussed this in 1.3, but we give more details in this chapter. In fact, a mobility means that an EU changes the outer IP address of the IPsec tunnel but the IKEv2/IPsec parameters remain in the same nodes. On the other hand, during a failover event, all the previous negotiated IKEv2/IPsec parameters must be transferred towards a new SG which ensures the continuity of a VPN session.Thus, the affected EUs attaches a new SG.

We concentrate on the methods that ensure IPsec availability as well as the continuity of an IPsec service. This mainly involves the transfer of a tunnel between different SGs. We also evaluate how an EU is affected when this happens.

The scenario we consider is an EU that wishes to reach an HTTP server placed within a trusted network protected by a SG. The EU first establishes a VPN towards a SG in the trusted network. Hence, the SG authenticates and protects the traffic between the EU and the HTTP server. We use strongSwan [16] to establish these tunnels. In order to provide HA capacities,

we use ClusterIP, allowing to build a cluster of SGs that share a common IP address without having a physical machine to perform this task. Thus, each SG determines from the incoming packets whether it is responsible for it or not.

4.2 Test-bed

Our testbed consists in a cluster of two SGs configured with a common IP address. When a EU establishes a VPN towards the cluster, one of the SGs is considered the active SG, whereas the other member is considered the passive (also known as stand-by SG). The active SG is the one that takes responsibility of the tunnel, whereas the passive SG is waiting to become the newly active SG if a failure occurs to the active SG. Both SGs synchronize all their IPsec tunnels so that they keep track of every single tunnel established with any EU. Our experiments are mainly focused in causing a failure to the active SG during a VPN session and evaluating the High Availability performances of the platform. At this point, the passive SG (not affected by the failure) detects no activity through the Synch Channel between them and consequently applies a new ClusterIP policy and becomes the newly active SG for a given IPsec tunnel. This ensures availability of the VPN services and avoids renegotiation from scratch of each tunnel of concerned EUs.

4.3 Position of our Work & Related Work

Concerning High Availability within IPsec, there exists similar works and approaches:

- **Yu [35]** proposes to solve availability issues on IPsec by simulating a cluster mechanism for IPsec gateways. As far as we know, this is the only article that addresses similar issues. Its *seamless switching* mechanism aims to spread SAs among both active and passive SGs. The author does not recommend a High-Reliable link between SGs in order to communicate the SAs, but the article mentions that the members of the cluster could be deployed in different network segments. The *seamless switching* process consists of adding a notify payload during the IKE_AUTH exchange in order to establish two tunnels (one VPN towards the active SG and a second stand-by VPN towards the passive SG of the cluster). The passive SG receives the IPsec information from the responsible active SG and installs a passive VPN towards the EU. Finally, there is also a mechanism of *seamless switching* to transparently change from the active-to-passive SG and to synchronize ESP replay sequence number. On the other hand, the case of a heavily loaded SG is not considered. The results are based on simulations and not in real implementation. By contrast, our

ClusterIP-based mechanism does not need additional IKEv2 payloads, so it is transparent to EUs.

- **RFC6027: IPsec Cluster Problem Statement [36]** unifies the terminology in terms of IPsec clustering. It also adresses the problem statement and the requirements to implement IKEv2/IPsec clusters. Multi-vendor solutions considering to implement IKEv2/IPsec clusters should use this terminology. This document identifies the main issues while implementing IPsec Clusters. These are the IKE Message ID and IPsec counters, a lot of long-lived state, the simultaneous use of IKE and IPsec SA among different members and the SPI allocation.These issues are explained in detail along this article. Finally, here are some definitions extracted from the document that helps to position our work:

- **RFC6311: Protocol Support for High Availability of IKEv2/IPsec [15]**. This RFC proposes an extension to the IKEv2 protocol. It aims to solve the refreshing of both IKEv2 (IKE_SA) and IPsec (CHILD_SA or IPsec_SA) counters due to a mismatch caused by a failure take-over process. The scenario addressed in the document is oriented to solve hot-standby IPsec-Clusters failures. A hot-standby IPsec-Cluster consists in a group of IPsec SGs in which there is only one member active at a time. All the IKEv2/IPsec contexts **are distributed among all the members of the cluster**. When a failure occurs on the active SG during an IKEv2/IPsec communication, the End-User (EU) continues sending IKEv2/IPsec traffic towards the cluster, leading to unsynchronized counters and packets loss. In order to reestablish the synchronization of counters, the new SG sends an IKEv2 message request of type INFORMATIONAL in order to negotiate counters. Concerning our work, this protocol solves the main troubles of synchronization of both IKEv2 messages (IKE_SAs) and IPsec counters (CHILD_SAs). Although it involves changes to the IKEv2 protocol, this extension handles the renegotiation of the IKEv2/IPsec counters in an efficient manner and should be considered if any IKEv2/IPsec counters mis-synchronization occurs.

4.4 IKEv2/IPsec and High Availability Constraints

IKEv2 and IPsec were primary designed for static configurations. The IKEv2/IPsec contexts have been designed to remain installed in the same device during a session or VPN tunnel establishment. However, todays requirements demand facilities to profit from mobility, handover, offload or even VPN session migration between SGs. This situation leads often to new extensions. For example: MOBIKE extension for mobility [3], MOBIKE-X draft for mobility and multihoming [32] [37] [38], CXTP and mechanisms to transfer security contexts [7] [11] [18] or a High Availability protocol to synchronize counters [15].

The huge demand on mobile data have increased the demand on system's availability. This

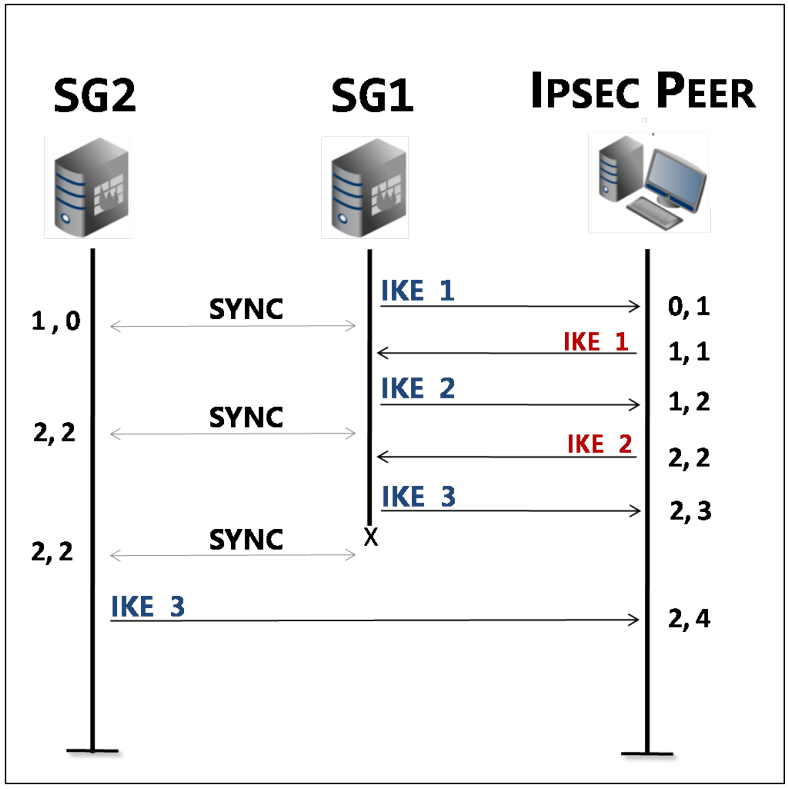
is the goal of ClusterIP. Clustering a VPN service at the network layer level (e.g. IPsec) seems to have positive impact when dealing with reliability of IPsec-based communications. Because IPsec imposes a strict check of its counters, the main issue is to synchronize both SG's IPsec parameters and counters. When an EU establishes an IPsec protected communication with a server, it first needs to configure how to protect the communication with the SG. The protocol used to negotiate the security parameters of the communication is IKEv2. Further exchanges allow to agree on a secure channel that serves to negotiate the IPsec parameters (to actually protect the IP traffic). IKEv2 and IPsec are **different** protocols with their **own** counters. The *message ID* counters provide anti-replay protection and keep track of every IKEv2 exchange, whereas the *sequence numbers* provide anti-replay protection and strict control of IPsec traffic. Even though the amounts of IKE messages do not represent as much traffic as the IPsec traffic, the synchronization of *messages ID* is critical due to the very strict control of the IKE_SAs when setting up a VPN tunnel and a very narrow window size.

4.4.1 IKEv2/IPsec Counter Synchronization

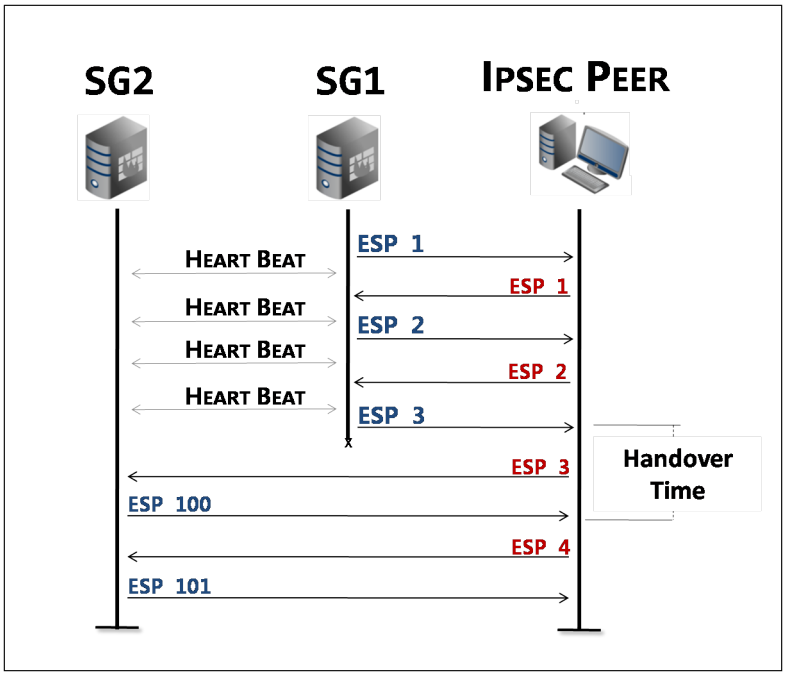
The IKEv2/IPsec suite aims to protect IP traffic. However, IKEv2 and IPsec act at different layers. IKEv2 is an application layer protocol which queries and responds to port 500 and 4500 in order to negotiate a secure channel between two endpoints sharing an IKE Security Association (IKE_SA). On the other hand, IPsec is a protocol that takes place at the IP layer and protects the traffic according to IPsec Security Associations (IPsec_SAs) policies, which are previously negotiated through the IKE_SA secure channel. All IKEv2 exchanges consist of a request-response pair of messages. It is mandatory to retransmit a request until it has been acknowledged. The IKEv2 window mechanism allows to send some IKEv2 requests without receiving a response, but once the window's limit is achieved, the oldest request MUST be acknowledged resulting in a window increase by one. This window is managed through the IKEv2 counters called: *Message ID*. Each IKEv2 message header includes its corresponding *message ID*, so that IKEv2 can strictly control the window as well as all unacknowledged messages. When the *message ID* is n and the window size is w, only IKEv2 messages between $n + 1 <$ *Message ID* $< n + w$ can be processed. If no acknowledgment is received after a long period of time, then the IKE_SA is deleted. Figure 4.1a shows an example of the mis-synchronization during IKEv2 exchanges between two endpoints with a window size $w = 1$.

Challenges concerning IKEv2 when implementing cluster of IKEv2/IPsec SGs are:

- *Stale Value of Message ID:* when takeover takes place, it is possible that the newly active SG is not aware of the last IKEv2 response made by the previously active SG. If this happens when taking responsibility, then the *message ID* used by the new responsible SG will be stalled. Figure 4.1a illustrates this case.



(a) Stale Messages ID values



(b) Stale Sequence Number value

Figure 4.1: IKE/IPsec counters desynchronization

- *Unacknowledged Request:* when a passive SG takes responsibility during a fail-over, it may happen that it is unaware of the last request sent, and thus the counter is stale. Receiving an unexpected *message ID* response would result in discarding the packet. This leads to IKE_SA destruction.

The anti-replay parameter of the IPsec Security Association is called *sequence number* counter. When the Anti-Replay service is enabled, it controls every incoming/outgoing IP packet protected with IPsec. Note that IPsec ALLOWS a sender to skip forward by sending a higher *sequence number*. Remember also that a duplicated usage of a *sequence number* is forbidden. Thus, when the *sequence number* counter is n and the window size is w, any message with *sequence number* $< n - w + 1$ will be discarded. A big window size (E.g. 1024) means that a node is capable to handle a bigger range of packets that arrive out of order. On the other hand,when *sequence number* $= n$ and the window size is very little (E.g. $w = 1$), the node is capable to remember only the last *sequence number*. The following packet MUST have a value *sequence number* $> n - 1$, otherwise the packet is dropped.

The biggest challenge concerning IPsec when implementing cluster of IKEv2/IPsec SGs is:

- *Stale Sequence Number value:* when a passive SG takes responsibility and becomes the newly active SG, it may happen that the *sequence numbers* are out of date. Figure 4.1b illustrates this situation. This occurs when the newly active SG starts sending IPsec protected packets with staled *sequence numbers*, implying that the EU rejects all the duplicated packets due to anti-replay protection of IPsec. Instead, note that IPsec allows to increase these *sequence numbers* without preventing the EUs, and the communication would not be interrupted.

There are three ways to avoid sequence number values. First, in the case that the newly active SG keeps an IPsec session from another SG, it may send an IKEv2 message to the EU in order to update both the Message ID and the sequence number value. However, this solution requires to create a new notify payload and thus to modify the IKEv2 protocol itself. Indeed, [15] proposes this method of resynchronization.

The second approach is a client unaware method. It consists in creating a cluster of IPsec SGs which maintain synchronized Message ID values as well as sequence number values. These latter can be synchronized by implementing ClusterIP (see section 4.5.2).

The third approach concerns the sequence number values only. When a newly active SG takes responsibility of a tunnel, it may skip a big number of sequence number values and the IPsec session will still not be interrupted. Note that the IPsec protocol allows both endpoints

to skip some sequence number values without prior agreement.

4.5 Clustering Methods for IPsec

This section positions Load-Balancing facing High Availability, details how takeover may impact the client and finally describes ClusterIP configuration and operation with IPsec.

4.5.1 Load-Balancing Versus High-Availability Clusters

A Load-balancing cluster is a set of nodes where more than one of the members may be active at the same time. Load-balanced clusters are implemented by sharing the workload between cluster nodes and offering better performance. HA-clusters operate by having redundant nodes intended to provide a service when other node fails. For Linux, the latter has been implemented using a free software package developed by the Linux-HA project, having the heartbeat software as the main product. Heartbeat automatically monitors resources so that they can be restarted or moved to another node on failure.

4.5.2 ClusterIP Implementation

When using a ClusterIP approach, no special hardware is required to benefit from Load-Balancing. Indeed, ClusterIP is intended to provide load-balancing features without having a load-balancer. The configured members of the cluster does share a multicast MAC address and thus receive the same packets. Then, a lower-layer mechanism (netfilter code) on each node, filters packets by calculating responsibility through an algorithm (E.g hashing the IP source of each packet). When applied, ClusterIP acts as a parameter for the `iptables` command.

The nodes in a cluster usually have two Network Interface Cards (NIC). One of the NIC's MAC address is replaced by the shared cluster MAC address and then a common virtual IP address is mapped onto it. The other NIC, being completely independent, can be used for any other purpose, as for example inter-nodes communications (e.g. heartbeat mechanism). Given the case where a machine counts with only one NIC, it is also possible to install a second virtual IP address on the same interface.

4.5.2.1 ClusterIP & IPsec

Originally, ClusterIP does NOT handle IPsec traffic. Aditionally, a given IPsec tunnel is associated to a single security gateway. Two SGs cannot handle the same IPsec tunnel unless the two SGs share a common IPsec context for the same communication. However, if an IKE daemon (e.g. strongSwan) handles to synchronize IKE_SAs and IPsec_SAs, a modified version

of ClusterIP that handles IPsec traffic could solve synchronization troubles; but the overhead for synchronizing ESP sequence numbers could be very high. Thus, deploying IKEv2 and IPsec in a cluster requires the synchronization of a large amount of information among all the cluster members. On the other hand, if less information is synchronized, fail-over would take longer to perform. As stated in 4.4.1, synchronizing counters might be the major barrier to overcome when it comes to setting up a cluster with IPsec. Some parameters involved in an IKEv2/IPsec session establishment are long lasting:

- *IKE Security Associations*: a SG may establish hundreds or thousands of IKE_SAs. Also, they may live for several minutes, hours, or days. They contain keys, selectors and other information concerning IKE traffic.
- *IPsec Security Associations*: a SG may establish hundreds or thousands of IPsec_SAs. Also, they may live for several minutes, hours, or days. They contain keys, selectors and other information concerning IPsec traffic.
- *Security Policy Database SPD*: they may live as long as an IKE_SA but they also tend to live longer in some operative systems.

IKE Counters (Message ID Counters) are the longest living states but at the same time are required to synchronize less often since synchronization might only occur whenever an IKE_SA is created or some INFORMATIONAL or REKEY exchange occurs. However, IKE needs to update the Message ID Counter immediately, as processing a message having a higher ID is not allowed (see 4.4.1). This is achieved by synchronizing IKE message counters after every single IKE exchange.

Concerning the anti-replay counters, every ESP/AH protected packet carries a *sequence number* that cannot be reused since the anti-replay feature would consider it as an attack, leading to drop all the packets and issuing attack warnings. Synchronizing anti-replay IPsec counters is not reasonable neither, due to the high load introduced (for each packet emitted). As a result, the designed solution synchronizes the counters every n-th packets (10.000 packets). This choice is justified since skipping *sequence numbers* is allowed in IPsec, and highly reliable delivery service (as in IKEv2) is not provided.

4.6 StrongSwan's ClusterIP-based HA Plugin

StrongSwan is a complete OpenSource IPsec-based VPN Solution for Linux operating systems. Its High-Availability plugin implements a ClusterIP-based mechanism that is able to maintain IKE_SAs and IPsec_SAs in case of failover. The current release strongSwan 5.x supports clusters

of two nodes maximum. This section explains how the HA plug-in achieves active/active High Availability and Load Sharing capabilities.

The IKE daemon of strongSwan synchronizes the IKE_SA database and the basic IPsec_SA database without *Sequence Numbers*. The remaining tasks are carried out by a modified ClusterIP plug-in called High Availability (HA). The HA plug-in requires two patches against the kernel in order to allow ClusterIP to work with IPsec. These patches modify the ClusterIP netfilter module, more specifically, the PREROUTING hook that marks received packets for forwarding before the decryption/encryption process. A third patch is required to modify the Linux firewall (iptables) in order to work over the patched kernel.

4.6.1 IKE Daemon Implementation

- *Daemon Hooks*: a hook is a function that is in charge of collecting information (in this case Synchronization data) for later use in preparing messages that are going to be sent to other members in the cluster in order to notify SA state changes or pushing information towards the plugin. Hooks are created and registered by the plugin at the daemon bus. Table 4.1 shows the hooks used by the HA plugin.

- *Synchronization messages*: table 4.2 shows the different synchronization messages types that can be exchanged between nodes in a cluster according to the implemented HA plugin of strongSwan. Messages are sent with no encryption by the hook functions using UDP datagrams on port 4150. However, an IPsec tunnel could be established in order to transmit this information.

- *State synchronization*: state changes are executed by Synchronization messages exclusively. They carry all the information required to create a duplicate of the active node IKE_SAs and IPsec_SAs. Duplicated IKE_SAs do not handle traffic and are installed in a PASSIVE state while duplicated IPsec_SAs are installed in the Kernel and subjected to ClusterIP algorithms.

- *Control messages* : table 4.3 shows the different control messages implemented by the HA plugin. These messages are sent along with the synchronization messages with the purpose of notifying segment responsibility changes.

- *Failover*: the state of a segment in a cluster is set by the HA plugin to either ESTABLISHED or PASSIVE. This is decided on each node using the same ClusterIP hashing function based on the source IP address. By using the same hashing function it guarantees that the cluster responsibility will not be assigned to both nodes at the same time. The activation/deactivation of a segment is performed over all the IKE_SAs that

Table 4.1: Hooks used by the strongSwan's HA plugin

Hook	**Description**
ike_keys( )	Receives IKE key material (DH, nonce, proposals)
ike_updown( )	Monitors state changes of IKE_SAs
message( )	Used to update IKE *Message IDs*
child_keys( )	Receives CHILD key material
child_state_change( )	Monitors state changes of IPsec_SAs

Table 4.2: Synchronization messages of the HA plugin

Synch Message	**Description**
IKE_ADD	Message used when a new IKE_SA is established. It contains all information to derive keys
IKE_UPDATE	Message used to update information of a concerned IKE_SA, for example, when authentication is done
IKE_MID_I	Updates the *Message ID* of the initiator
IKE_MID_R	Updates the *Message ID* of the responder
IKE_DELETE	It is used to delete a corresponding IKE_SA
CHILD_ADD	Message used when adding a new IPsec_SA
CHILD_DELETE	Message used to delete an IPsec_SA

are found on that actual segment. There is no impact on their IPsec_SAs, they are always active.

- *Node reintegration*: reintegration is meant to take the failed node after its recovery and reinserting it into the cluster as a backup node again. The recently reincorporated node needs to fully synchronize the state information; this is achieved by pushing all the active IKE_SAs messages, cached in the active node, onto the newly arrived node. Synchronizing IPsec_SAs is not possible using the cache, as the messages do not contain *Sequence Number* information managed in the kernel. To reintegrate a node, the active node initiates rekeying on all IPsec_SAs.

Table 4.3: Control messages for segment changes notification

Control Message	**Description**
SEG_DROP	Message to drop responsibility of segments
SEG_TAKE	Message to take responsibility of segments
STATUS	Heartbeat mechanism to prove liveness and segment responsibility
RESYNC	Used to resynchronize a list of segments

4.7 Performance Tests & Results

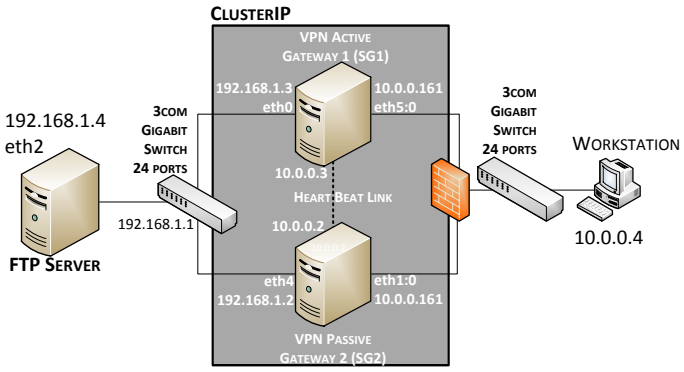
4.7.1 Testbed description

Our performance tests are conducted in two different topologies, one with HA-plugin enabled (i.e. based on ClusterIP) and the second one with no HA features (i.e. no ClusterIP) to compare how ClusterIP improves the performances. The first scenario, shown in figure B.2a, counts with an HTTP server, an IPsec peer and two VPN SGs; strongSwan is configured to provide High-Availability cluster between the SGs and its members keep in synch through the Heart Beat link (Synch Channel). In the second topology, illustrated in figure B.2b, the whole traffic goes through a single active SG, meaning that strongSwan is used as a VPN solution but with no fail-over node, thus the HA plugin is not loaded.

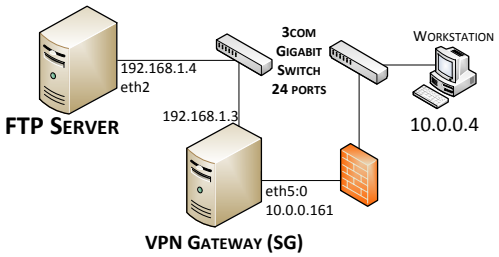
The results are represented in graphs with box-and-whiskers style [39] (refer to section 3.4.3 for more details). As stated before, this kind of representation is used to plot statistical data. The box-and-whisker plot indicates: (i) the smallest observation, (ii) the lower quartile, (iii) the upper quartile, (iv) the largest observation and (v) the median. The space between the lower and upper quartile represents 50% of the samples.

During the tests, we used two different implementations (`time` and `top`) in order to measure the IPsec performance under different circumstances. The command `time`, launches the specified program command with given arguments. When the command has finished to run, `time` writes a message to the standard output giving timing statistics about this program run. The outputs of `time` are (i) the *elapsed real time* between invocation and termination of the command, (ii) the ***user CPU time*** or ***cpu-us*** spent executing instructions of the calling process and (iii) the ***system CPU time*** or ***cpu-sys*** spent in the system while executing tasks on behalf of the calling process.

The command `top`, provides the real-time CPU activity. It shows a list of the ongoing system tasks . We identified the IPsec related task ID within this list and we collected all the information concerning the CPU consumption during the tests. All tests were done using different number of CPUs in order to compare the impact of having several CPUs sharing the workload (1,2,3 or 4 CPUs).



(a) Scenario using ClusterIP



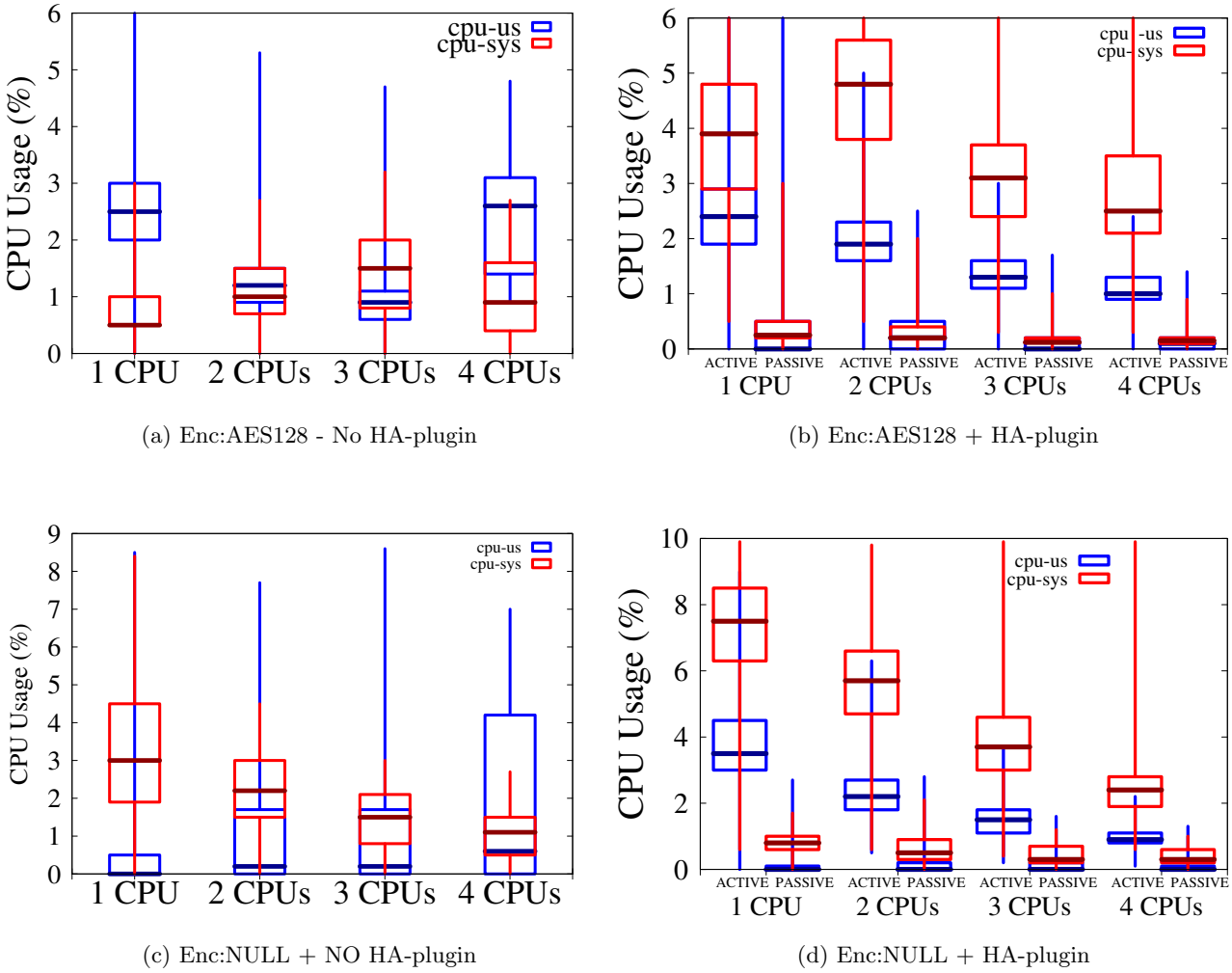
(b) Scenario NOT using ClusterIP

Figure 4.2: Scenarios

4.7.2 First Test - ClusterIP overhead Measurements Test

The purpose of this test is to evaluate how much CPU resources and time the modified ClusterIP adds to the whole VPN service. Note that when the HA plugin of strongSwan is activated (i.e. B.2a), for each incoming packet the modified ClusterIP hashes the IP header to check whether or not the SG is the responsible node to handle that packet.

This test is performed using both topologies described in figures B.2a and B.2b, thus comparing the impact of ClusterIP. The test consists of downloading a file of $1GB$ from an HTTP server (placed behind the SG) towards the peer by using the command `wget` on the peer's side. During the tests based on figure B.2a, the HA plugin is enabled whereas during tests as in figure B.2b the HA plugin is deactivated. We measured the CPU impact and the time spent by the system to complete an HTTP download (using the `wget` command). The IKEv2 exchanges are always protected with AES128-SHA1, whereas two different algorithms for ESP encryption were also analyzed throughout the test: AES128-SHA1 and NULL-SHA1 (where NULL means no encryption, note that strongSwan does not support no integrity check and it is always SHA1). We also varied from one (1) to four (4), the number of CPUs available on each SG; this allows analyzing the evolution of the CPU consumption of both types of encryption. As mentioned in 4.7.1, the CPU consumption and the download time (user and system time) are obtained through the `top` and `time` command respectively.



(a) Enc:AES128 - No HA-plugin

(b) Enc:AES128 + HA-plugin

(c) Enc:NULL + NO HA-plugin

(d) Enc:NULL + HA-plugin

Figure 4.3: First Test: Download Performance Test (CPU Usage)

CPU Consumption: figures 4.3a and 4.3c represent the CPU usage with no HA features. They implement different encryption algorithms. Figure 4.3c uses NULL encryption whereas figure 4.3a uses AES128. When NULL encryption is used, packet treatment is improved and the CPU consumption decreases around 30% . The CPU consumption of the userland is practically the same for both cases. So, NULL encryption might improve CPU performance but it also might downgrade the security level as the flow is integrity protected but not encrypted.

Considering AES128 and NULL encryption but with HA-plugin activated, figures 4.3b and 4.3d illustrate the CPU consumption and show with different active CPUs a 3 to 8% CPU Usage. AES128 registers more activity (3% to 8% of CPU Usage) than a node using NULL

encryption (2.5%to 5% of CPU Usage). Once again, the CPU consumptions of the userland have the same behavior for both scenarios. As in AES128 with no HA-plugin, a peak is observed in the case where 2CPUs are used. We observe that the ClusterIP-based plugin is not well suited when 2CPUs are being used.

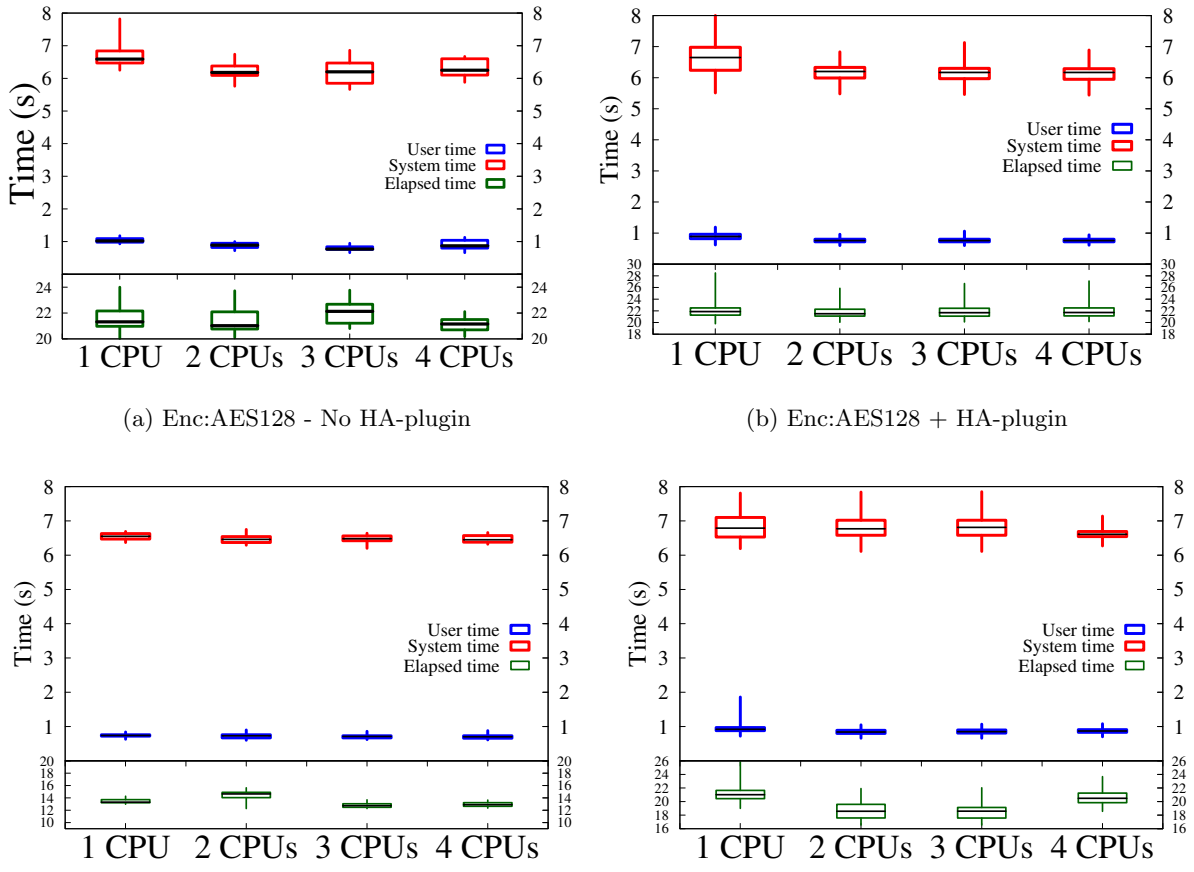
Download Time: Results concerning the HTTP download time are shown in figures 4.4a, 4.4b, 4.4c and 4.4d. Two main scenarios are compared: the impact of HA-plugin on the download time and the impact of the encryption method used during these downloads. When no HA-plugin is used, AES-128 encryption introduces more System Time than when NULL encryption method is used. The System Time and the Elapsed Time (total time to complete the download) increase by 11% using AES128. However, no variation is observed in the User Time, which means that the operating system always performs the same user mode tasks. Note that the modified ClusterIP requires the kernel to be patched in order to allow IPsec packet filtering. No variation of the User Time was observed when using the HA-plugin. It always stayed around $1s$ in all scenarios of the first test. Finally, when comparing the impact of the encryption methods showed that a cluster with AES128 takes around 30% more time to download than a download that uses NULL encryption.

4.7.3 Second Test - QoS on an HTTP connection

The second test evaluates the quality of service (QoS) ensured by the HA plugin in terms of upper-layers (e.g. HTTP-based downloads) reactivity. The test consists of downloading a file of size 50MB from the HTTP server towards the VPN peer. On the client side, we measure the Elapsed Time (obtained via the command `time`), which represents the time to complete the download. We do the same for both topologies (with & without HA-plugin, figures B.2a and B.2b correspondingly), thus comparing the impact of the HA-plugin and added overhead during the download.

Both scenarios consider the encryption and integrity protection with AES128 and SHA1 algorithms respectively. With the ClusterIP-based plugin activated, both figures illustrate the time to download a file of size $50MB$. Figures 4.5a, 4.5b show the download time through Ethernet connections. The User Time stays invariable.

On the other hand, differences are observed when the HA-plugin is activated. The fact to decide either to treat or not a packet (by filtering at the IP layer), increases the System Time by 35%. Also the Elapsed Time increases by 25%, which introduce some overhead.



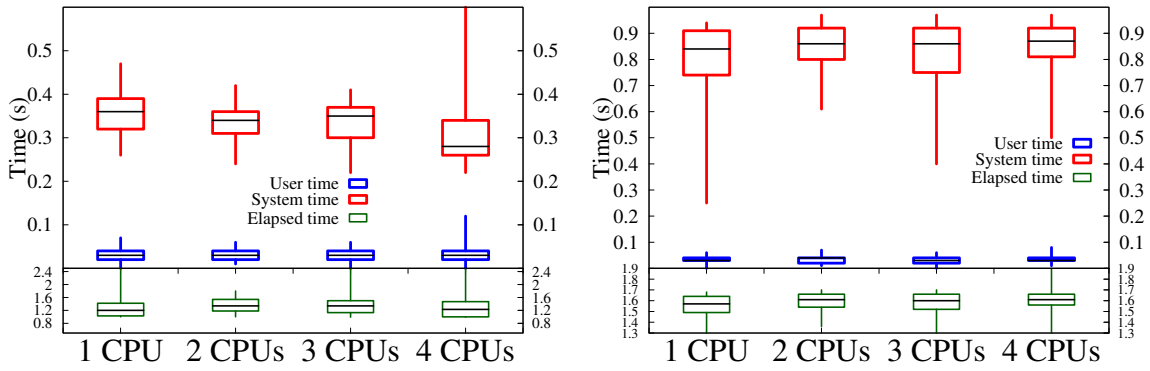
(a) Enc:AES128 - No HA-plugin

(b) Enc:AES128 + HA-plugin

(c) Enc:NULL + NO HA-plugin

(d) Enc:NULL + HA-plugin

Figure 4.4: First Test: Download Performance Test (time)



(a) Enc:AES128 + NO HA-plugin

(b) Enc:AES128 + HA-plugin

Figure 4.5: Second Test - QoS on an HTTP Connection

Throughout this test, the fact of using 1,2,3 or 4CPUs did not impact noticeably the download time. For example, figures 4.5a and 4.5b show that the behavior of the download time is very similar for each CPU scenario. Finally, as we can see, upper-layers might be impacted when this HA feature is activated. Administrators wanting to implement a cluster of VPN SGs should justify the need for this solution. A typical example is an environment where thousands of tunnels should be established, so letting the cluster to spread responsibility over different nodes, and thus spreading the load on each SG. Even if the load for a single connection would be slightly higher by using the HA-plugin, a real positive impact would be noticeable only when a large number of VPN connections are spread among the cluster members.

4.7.4 Third Test - Interruption Time of an HTTP communication

When a failure event occurs, the ClusterIP module used by strongSwan allows the cluster to switch transparently from the active node to the passive SG node. We concentrate on evaluating the handover network time (see figure 4.1b) during fail-over. The scenario is illustrated in figure B.2a. The test consists of downloading a file of size $500MB$ from the HTTP server towards the VPN peer. After 5 seconds of download, we interrupt the outgoing network interface of the active VPN SG, causing a failure event. The passive SG of the cluster stops receiving responses through the Synch Channel, and thus takes responsibility of the VPN tunnel. We captured the traffic between the VPN peer and the cluster during the test. Figure 4.1 represents some protected ESP traffic at the IP layer. The period of time between the last ESP packet emitted by the active SG (SG1) and the first ESP packet emitted by the passive SG (SG2) is considered as the *Handover Time*. This time corresponds to the network delay of takeover from the active node to the passive SG.

Figure 4.6 represents the results obtained when measuring the *Handover Time*. The quartiles

illustrate the download time. The black colored quartiles shows the download time during a fail-over event, taking $3-4$ seconds more than a download without interruption (colored in red).

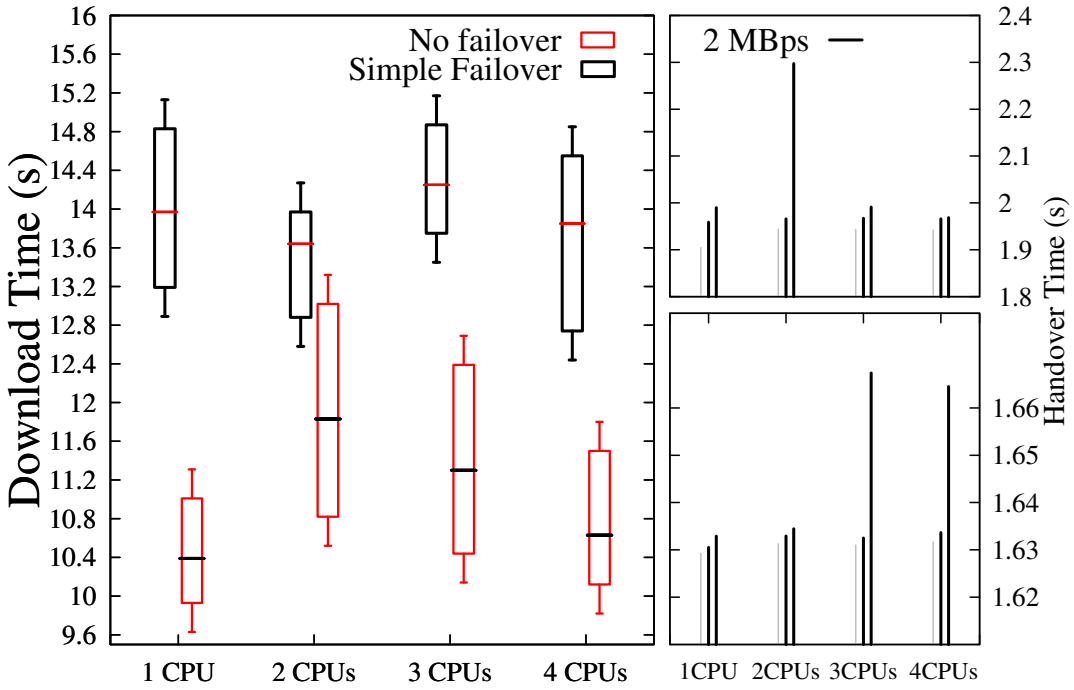
Note that the handover time (at the right-bottom and right-upper side) are illustrated with three vertical lines that represent the lower quartile, the median and the upper quartile. The handover time with no traffic control varies from $1.63s$ to $1.66s$ (right-bottom figure). This time is considered as network-friendly time to perform a fail-over. However, the difference between the download time on both scenarios (with and without fail-over) stays over $3s$, $1.7s$, $3.2s$, and $3.1s$ for 1,2,3 and 4CPUs respectively. This overhead or difference is due to updating the IPsec databases. An update action is blocking the database and thus blocking the communication. This delay is expected to increase as the number of tunnel increases. Upper-layers treatment add more delay to the communication as well the system also takes some time in order to accomplish all tasks. The results are compliant to the expected $1s$ to $3s$ performances as required in strongSwan's specifications. We also tested an additional *Handover Time* where the bandwidth limit is imposed to $2MBps$, emulating the use case of a RAN (Radio Access Network). Once again, the network fail-over time remains between $1.9s$ and $2.3s$. The reason is that with reduced rates, the platform is not overloaded. The impact of the number of CPUs cannot be measured because, the time between heartbeat exchanges are dominant. Nevertheless, the *No Failover* case (colored in red), shows that the ClusterIP module has better performance when 1CPU or 4CPUs are being used. In terms of QoS, it should be considered only to use 1CPU or 4CPUs , instead of 2CPUs or 3CPUs.

4.8 Conclusions

Throughout this chapter, we measured the impact and performance of using a modified ClusterIP module in order to clusterize SGs over IPsec. The availability of the IPsec service is improved thanks to the hot-standby clustering ensured by ClusterIP. Also, *strongSwan* guarantees the continuity of an IPsec session thanks to its HA plug-in allowing to spread all the IPsec tunnels among its cluster members. Results showed that an active SG spends 5% to 8% of CPU more than a passive SG when clustering IPsec tunnels. We also observed that there is an additional load when using the HA-plugin of strongSwan among the cluster members. Downloading a $500MB$ file takes 20% more time when using ClusterIP and the HA plug-in. As such, an administrator willing to implement this solution should take into account the performance costs of adding HA features to its VPN service.

Furthermore, one main drawback of ClusterIP is its limitation to be deployed within a same network segment. Chapter 5 will concentrate on using our own VPN tunnel management tool

Figure 4.6: Third Test: Handover Time



that allows IPsec transfer between SGs within Mono-LAN environments. Chapter 6 will consider transferring an IKEv2/IPsec context between two SGs owning different IP addresses.

Chapter 5

Mono-LAN Context Transfer: Context Management

This chapter addresses the IKEv2/IPsec context transfer between different SGs within a Mono-LAN environment. Instead of using the HA plugin of strongSwan based on ClusterIP, we consider using our own developments. We manage the VPN connections dynamically through GET and PUT functions (introduced in section 3.4). As such, we explain how to perform GET and PUT between two different machines in a Mono-LAN environment.

We evaluated the impact on upper-layers when moving an IKEv2/IPsec context from one SG to another with the same IP address during an active IKEv2/IPsec session. First, we introduce our own HA module under Mono-LAN environments. Then, we concentrate on our developments allowing to dynamically move a VPN tunnel between different SGs (referred to as *Context Management* introduced in 1.2.2).

Notice that, as we address a Mono-LAN environment again, the IP address remains the same while transferring an IKEv2/IPsec context. Thus, the transfer is also transparent to End-Users. As such, the transfer is perceived as an interruption of the communication and no signaling is needed between the End-User and the SG.

5.1 Motivations and Mono-LAN Context Management

The management of large VPN clusters requires the conjunction of different mechanisms aiming to provide a fluid VPN service. First, the load balancing features are present when the VPN sessions are spread among different cluster members based on a predefined algorithm or some trigger event (i.e. overloaded SG). However, the cluster members must be able to communicate and synchronize the information and parameters of those VPNs sessions needing to switch to another equipment. Second, the High Availability features are present when some SG within the

cluster fails and some other SG (or several SGs) assume responsibility of the affected connections concerned by the failure. This might result in a re-organization of VPN sessions among the other cluster members. Nevertheless, this is also subject to an algorithm that decides whether a VPN connection might be handled by a specific SG or not (i.e. closest SG, better QoS, lower pricing rate, etc.).

In chapter 4, we observed that the strongSwan's plugin (based on ClusterIP) provides HA and load-balancing features but no context management facilities. On the other hand, as ClusterIP spoofs ARP requests, it is limited for deployment within a same network segment. This limits the deployment of VPN clusters where the members are separate geographically.

We intend to show why it might be advantageous to be able to migrate a VPN session of node from one SG to another. For example, considering that the node might be attached to a SG that is getting overloaded, it could be an advantage (in terms of QoS, pricing, latency, etc.) to attach a new SG that can deliver the same service without the risk to loose connectivity. This defines actually the idea of building a cluster of SGs that are strategically configured to improve the performance of the VPN service.

5.2 Scenarios: HA and Context Management

Following sub-sections 5.2.1 and 5.2.2 describe the scenarios addressed throughout this chapter. First, sub-section 5.2.1 presents the *High Availability* scenario, where two concepts are introduced: the heartbeat and the Sync daemons. We illustrate the cases where a VPN cluster is composed of n number of nodes and where $n = 2$. Second, we present the scenario of *context management*, where a VPN client can be attached to a new SG without any IKEv2 signalization.

5.2.1 High Availability for n gateways

Our first scenario is shown in figure B.3. It illustrates a cluster of n SGs configured to offer High Availability features. An EU wishing to reach a server (i.e. streaming server, web server or some other service) should first establish a secure connection with the VPN cluster, authenticating and encrypting the communication between them, as well as protecting the data flow that reaches the server containing the desired service. Initially, the cluster decides which SG is taking responsibility of the VPN session, and then the VPN tunnel is established. As we address a Mono-LAN environment, the EU attaches a unique VPN cluster's IP address. Some protocols or network configurations allow to maintain a shared IP address among different cluster members (e.g. ClusterIP, VRRP, or simply deactivation/activation of the network interface). Additionally to the IKEv2/IPsec instance (i.e. strongSwan), two independent modules ensure the HA features:

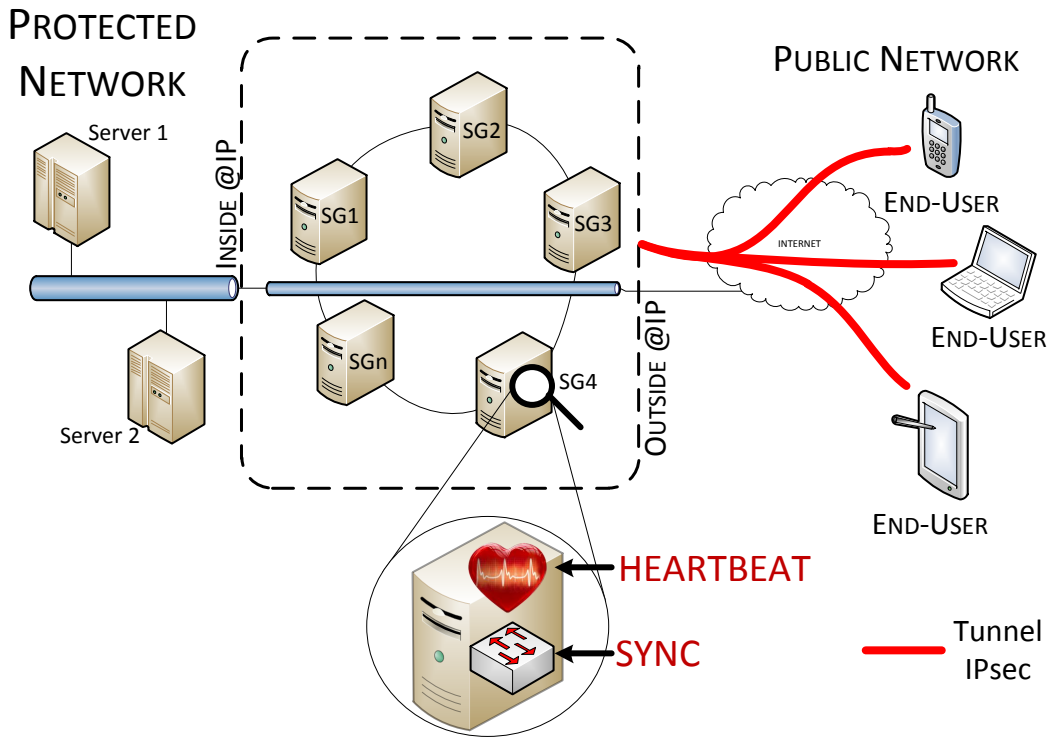


Figure 5.1: High Availability scenario for n gateways

- **Heartbeat** is a daemon providing cluster infrastructure services by ensuring aliveness of other cluster members. If any of the cluster member fails, the heartbeat allows detecting this situation.

- **Sync** is a daemon that provides synchronization of all the IKEv2/IPsec information. When an EU establishes a VPN tunnel towards the cluster, this information must be replicated to all the other members. The replication of the VPN session can be maintained locally within each SG or centralized within an independent device. Thus, the transfer of the IKEv2/IPsec sessions is needed to maintain the database up to date.

When both the heartbeat and the sync daemons are activated, the cluster is also capable to overcome failures. This is actually defined as fault tolerance. As such, if some failure occurs to the SG taking responsibility of some VPN session, then some other SG within the cluster must keep the VPN tunnel alive. This event is transparent for the EU and is interpreted as an interruption of the communication. No signaling is expected to happen, unless a refresh of the cryptographic material is desired.

High Availability for two nodes (*n=2*)

When the cluster is composed of two nodes (which is actually our testbed configuration in section 5.4.1), we consider that only one SG is active at a time (a.k.a. active SG), whereas the

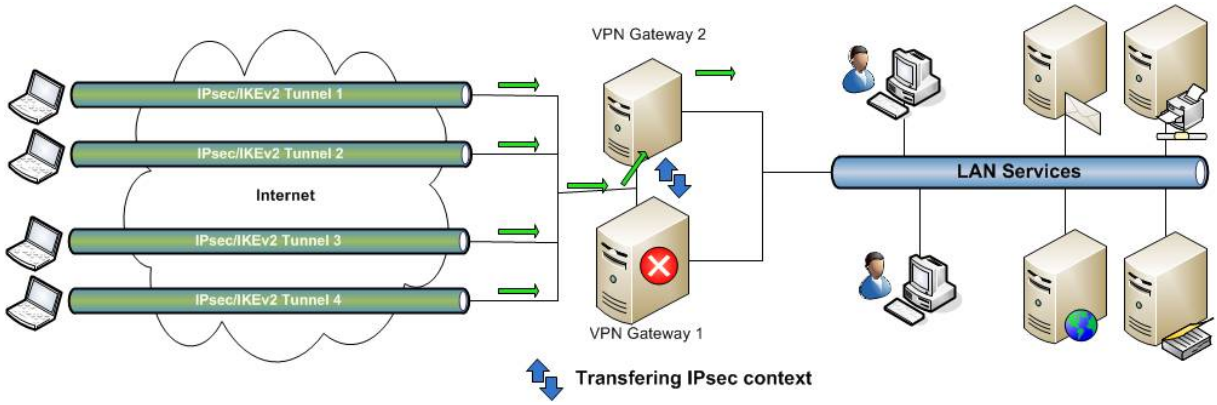


Figure 5.2: High Availability scenario for *n=2* nodes

other SG is considered as standby SG (which may become active whenever the first one fails). The active cluster member synchronizes every VPN session with its neighbor. This ensures perfect timing and synchronization of all the active VPN tunnels in the cluster.

As in section 5.2.1, we activate the *heartbeat* and a *sync* daemons which both ensure a keep alive mechanism together with IKEv2/IPsec state synchronization (including IKEv2/IPsec counters).

Figure 5.2 represents a cluster of *n=2*, where only one SG is active at a time. The SG1 (*VPN Gateway 1*) is considered as the initially active SG and is responsible for all incoming requests. Suddenly, some failure occurs in SG1 and then the SG2 becomes the active SG, taking responsibility of all the VPN connections. In this scenario, all the IKEv2/IPsec contexts that are stored in SG1 are also replicated on SG2 through the *sync* daemon (introduced in section 5.2.1). As such, when switching from SG1 to SG2, the VPN cluster offers fault tolerance and ensures the high availability of the VPN service.

5.2.2 Context Management

The second scenario considered is called context management (see figure B.4). The context management scenario includes an End-User that demands some service placed behind the cluster of SGs. The EU establishes a VPN tunnel in order to access a service securely (i.e. streaming video, mail server, web server, etc.). However, in some cases, the SG where a client is attached can get overloaded. The risk to loose connectivity with the SG2 is high. Some other SG (i.e. SG3) offers better performance to the EU than the initial SG2. It might be advantageous in terms of QoS, reachability, latency, among other reasons, to transfer the IKEv2/IPsec context from SG2 towards SG3. We intend to dynamically manage the VPN sessions between different SGs, allowing transparent context management. As the IP address of the cluster remains the same in Mono-LAN scenarios, no signaling is expected to happen among EUs and the SG. As shown in

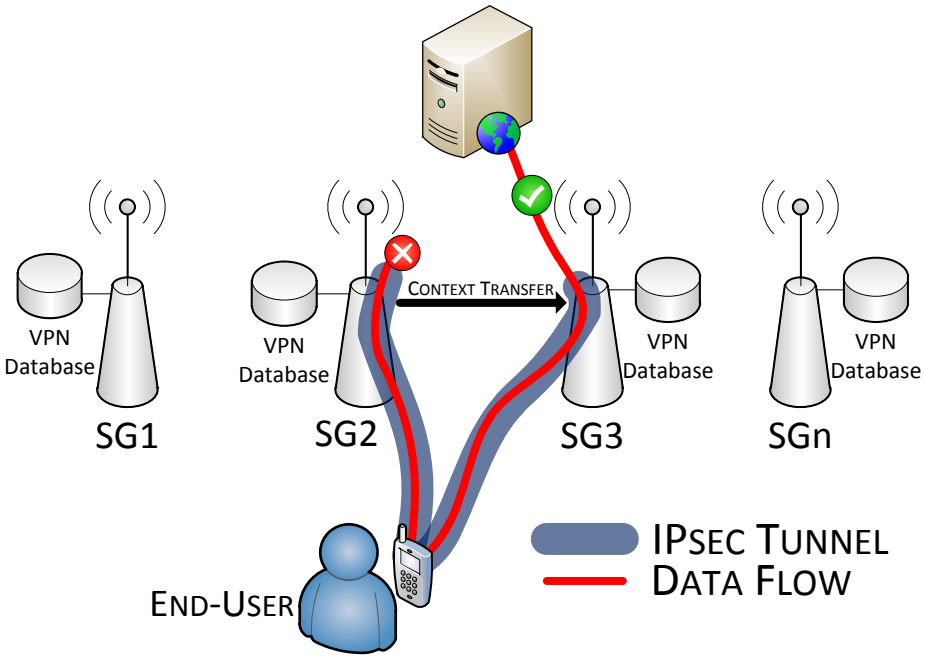


Figure 5.3: Context Management scenario

figure B.4, the IKEv2/IPsec context is transferred from SG2 to SG3 and all layers above benefit from this. Thus, no signaling is needed and this transfer is transparent for upper layers. This is only interpreted as some interruption or network delays. This avoids re-authentication towards the new SG, which in our example is SG3.

5.3 Position of our Work

Considering a Mono-LAN architecture, we positioned our work with some other protocols and documents that address similar topics. First we introduce an IETF document (RFC3374), which describes some reasons to perform a context transfer (regardless of the type of context transfer: IPsec, TLS, IP, etc.). Then, another IETF document (RFC6311) presents some recent investigations addressing the re-synchronization of IPsec counters for High Availability purposes. Finally, we present VRRP, a protocol offering similar solutions to ClusterIP (discussed in section 4.5), but it also defines a framework to set master/slave routers to provide fault tolerance features.

- **RFC3374 - Problem Description: Reasons for Performing Context Transfers Between Nodes in an IP Access Network** [40]: This Informational IETF document expresses the needs for transferring a service-candidate context and illustrates the overview of the reasons to perform a transfer of context between nodes in an IP Access Networks. It is often the case that a peer connected to a router, establishes a context of a service

that is provided by the access network. Usually, this service provides access to a subnet just like it is the case of IKEv2/IPsec protocol. The set of parameters to establish the connection between a peer and the router are the security associations. This information can be transferred from one router to another in order to boost the re-establishment of the connection when the IKEv2/IPsec Security Association is transferred between routers. However, there are advantages and drawbacks. The main advantage is the possibility of a fast service re-establishment (i.e. AAA, QoS, IKE Authentication, etc). However, one of the drawbacks is the compatibility between different routers. On the other hand, they might be situations where the access networks (or even the routers) would prefer to initialize a new session from scratch. In our case, for an IPsec environment, the established security associations between the node and the SG are supposed to be static. Nevertheless, new challenges like mobility, context transfer, multihoming, among others, concern IPsec. This causes some issues due to its static design. Properties like load sharing, load balancing, context management and high availability, are some of the reasons to justify the transfer of an IKEv2/IPsec context between SGs.

- **RFC6311- Protocol Support for High Availability of IKEv2/IPsec** [**15**]: as discussed in section 4.4, one main obstacle to perform HA or context management under IPsec environments are the counters associated with every security association established. This standard defines an extension of IKEv2 in order to allow resynchronization of those counters. There are actually two types of them: *message ID* and *sequence numbers* (see section 4.4 for details). When transferring the IKEv2/IPsec context between different SGs, we have taken into account the solution proposed by this standard: move the *sequence numbers* forward, for example, by adding 10.000 to the sequence number value, avoiding anti-replay protection delays. Actually, the IPsec standard does not forbid the fact to raise at some point the value of the *sequence numbers*. For example, when transferring a VPN session from one SG to another, it is almost sure that some packets treated by the new SG will be stale. These packets will be considered by the End-User as a replay attack and will be dropped. If these packets had higher sequence number, then they would have been decrypted/authenticated by the End-User normally. As such, each time we perform an IKEv2/IPsec context transfer, we rise the *sequence numbers* by a value of 10.000, avoiding stale counters.

- **RFC3768 - The Virtual Router Redundancy Protocol (VRRP)** [**41**]: intends to add fault tolerance of a single point of attachment in a network. VRRP creates a group of SG under a same virtual IP. Thus, the whole set of machines are acting as a single SG. Every interface that is configured with VRRP owns a virtual IP address that is common to all routers being part of the redundant topology. In the case where two or more SGs are configured with VRRP, the responsibility is determined by parameter

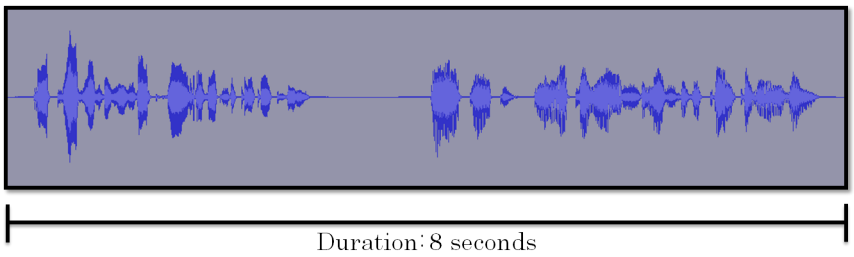


Figure 5.4: Original audio file transmitted on streaming during tests

called **vrrp-priority**. It defines which SG has the highest priority for getting responsibility among the group. The goal is to simulate a default SG with a unique IP address so that any End-User acceSSING through one of the SG does not know that there are many SGs composing the cluster. The responsible SG acting as default SG is called *master virtual router*, whereas all the other SGs are called *backup virtual routers*, which are ready to get the role of *master virtual router* and to forward packets if the *master virtual router* ever fails. Note that VRRP does not aim to perform load-balancing as it does not distribute the load of VPNs among different routers, but it adds fault tolerance (redundancy) to the network. In contrast, ClusterIP is able to spread the load among different SGs based on the hash of some input information (i.e. source IP address, destination IP address, port number, etc.). This permits VPN distribution among different SGs within a cluster, even though synchronization of the IKEv2/IPsec context must take place independently in order to ensure continuity of the VPN service. If we ever had to decide to use either VRRP or ClusterIP, we would most likely prioritize ClusterIP because it allows some sort of load sharing. However, the main drawback of ClusterIP is its limitation to be configured within the same network segment, disturbing the deployment of geographically distant SGs.

5.4 Performance Tests

5.4.1 Testbed description and scenarios

All our measurements are performed over an active VPN session during an audio streaming transmission with a duration of 8 seconds. Figure 5.4 illustrates the original audio file that is transmitted during tests. The signal contains a sentence pronounced by a woman and man. It is sampled at 48 kHz and contains up to 14 kHz signals.

We wish to measure the impact that HA and context management scenarios have on a VPN session. We consider different bit rate transmissions when streaming at 8Kpbs, 48Kbps and 96Kbps, evaluating the impact on the QoS (described in section 5.4.2.3). Additionally, we measure the impact of using two different encryption methods: CBC and CTR (described in

Architectures	Protocol	Encryption algorithms	Transmission bit rate	Frequency of interruptions [$times/sec$]
High Availability **(HA)**	RTSP (UDP)	AES-CBC	$8kbps$	f=$^0/_{8s}$
				f=$^1/_{8s}$
			$48kbps$	f=$^2/_{8s}$
Context management **(CxtM)**	HTTP (TCP)	AES-CTR		f=$^3/_{8s}$
			$96kbps$	f=$^4/_{8s}$

Table 5.1: Architectures and parameters addressed during tests

section 5.4.2.2). We also consider different transmission protocols: HTTP and RTSP, based on TCP and UDP respectively (see section 5.4.2.1 for details). Finally, we vary the frequency of HA and context management events. A switching frequency of f=$^1/_{8s}$ means that only one event takes place during the $8sec$ streaming, f=$^2/_{8s}$ means two events,f=$^3/_{8s}$ means three, and so on.

Table 5.1 summarizes the set of architectures and parameters for all our tests. The scenario we are considering is: initially, an EU establishes a VPN towards the cluster (composed of two nodes). This allows the EU to reach the streaming server in a secure manner. Then, the EU receives an audio streaming with a duration of $8sec$. Depending on the scenario, our performance tests are conducted either with HA features enabled (heartbeat and sync daemons activated as in section 5.2.1) or with context management features enabled (as in section 5.2.2). Once the streaming has finished, the original source file is compared with the received audio file recorded during streaming in order to measure the QoS by using the software POLQA (see 5.4.2.4).

- **First testbed and scenario - HA**: our first testbed, shown in figure 5.5, represents the logic topology of the HA cluster. This testbed includes an audio streaming server, an IPsec peer (EU) and two VPN SGs (SG1 and SG2). The *sync* daemon synchronizes every IKEv2/IPsec context established on both SGs and the *heartbeat* daemon is responsible for checking liveness of each SG every second. We focus on how to ensure fault tolerance and high availability features to EUs. Thus, during the audio streaming transmission, we simulate failures and vary the frequency of interruption $^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$) events on the cluster. This automatically activates the mechanisms to ensure continuity of service of the VPN service and makes the standby SG becomes the active SG and vice-versa. For experimental purposes, we have synchronized one VPN session.

- **Second testbed and scenario - context management**: the second testbed is shown in figure 5.6. It represents the logic topology of the testbed and also includes an audio streaming server, an IPsec peer (EU) and two VPN SGs (SG1 and SG2). We perform manual switching of a VPN session by transferring the IKEv2/IPsec context from one SG1 to SG2 (and vice-versa) using different frequencies ($^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$). The objective is to successfully manage an IKEv2/IPsec context between SGs at anytime during an

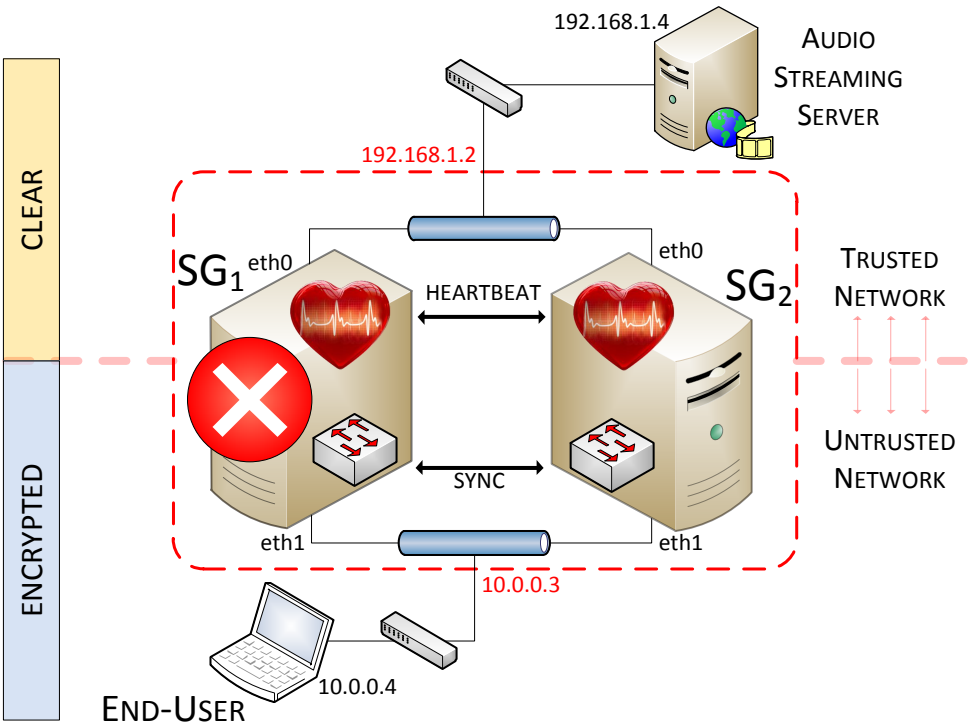


Figure 5.5: Testbed 1 - High Availability

audio streaming over an active VPN session, evaluating the impact over the IPsec layer and upper layers, as well as to provide some performance measurements during the audio transmission.

5.4.2 Protocols, parameters and audio streaming tools

This section describes in detail the different methods, protocols and bit rates used during our performance tests. We compare the difference and introduce the impact that they may have when securing with IPsec. The following list shows the different parameters, protocols and tools used for the tests:

- Protocol for audio transmission: HTTP vs. HTTP
- Encryption algorithms: AES128-CBC vs. AES128-CTR
- Audio tools supporting several bit rates: 8kbps, 48Kbps or 96Kbps
- Software for measuring audio QoS: POLQA

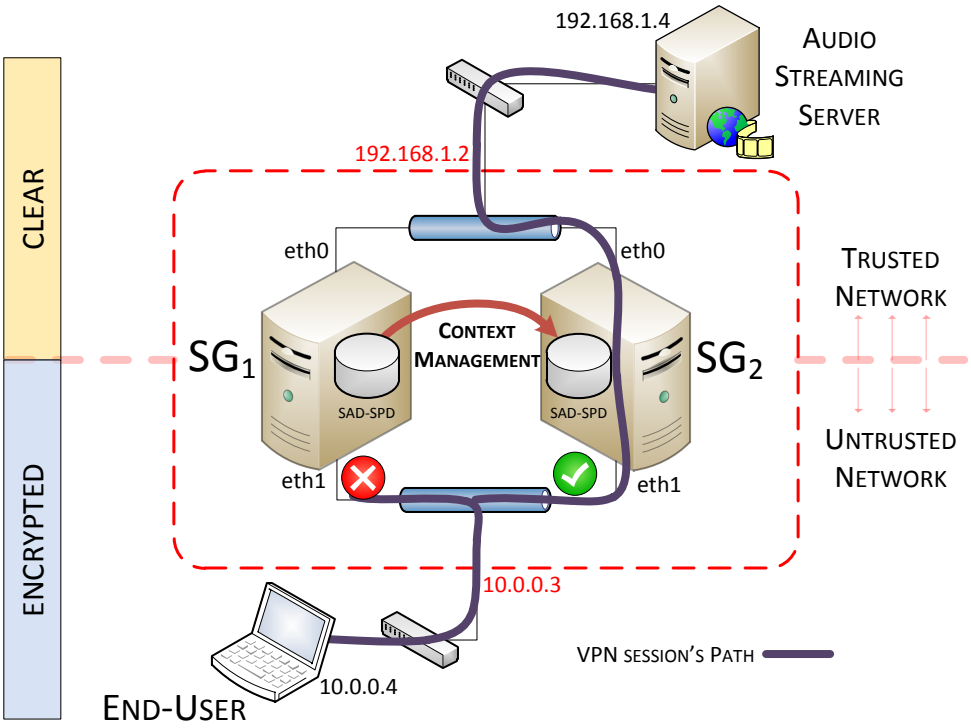


Figure 5.6: Testbed 2 - Context Management

5.4.2.1 Protocols for audio streaming transmission: RTSP vs. HTTP

We have considered two protocols for the audio streaming transmission, RTSP and HTTP. RTSP stands for Real Time Streaming Protocol, whereas HTTP stands for Hypertext Transfer Protocol.

This section is intended to explain how the audio streaming data flow is built, based on RTSP or HTTP, and protected with an IPsec environment. Figure 5.8 shows the packet format when securing a RTSP audio streaming (over UDP) with IPsec (ESP) in tunnel mode (refer to section 2.3.2 for details). On the other hand, figure 5.7 illustrates the packet format when performing HTTP audio streaming (over TCP) secured with IPsec (ESP) in tunnel mode. The RTSP and HTTP payload length varies depending on the audio source data itself. For example, during silences in the audio streaming, the size of the packets may be shorter than during voice transmission. However, this also depends on how the data flow is treated by the transport layer. This is discussed in more details in section 5.5.

Regarding the IPsec layer, the ESP header has a 4-byte length Security Parameter Index (to identify the corresponding security association) and a 4-byte length sequence number, which is part of the anti-replay protection of IPsec, avoiding duplicated packets with the same sequence number during ESP authentication. Although some encryption algorithms do not need padding to cipher a plaintext (e.g. CounTeR), ESP does require padding to 32-bit word-align the

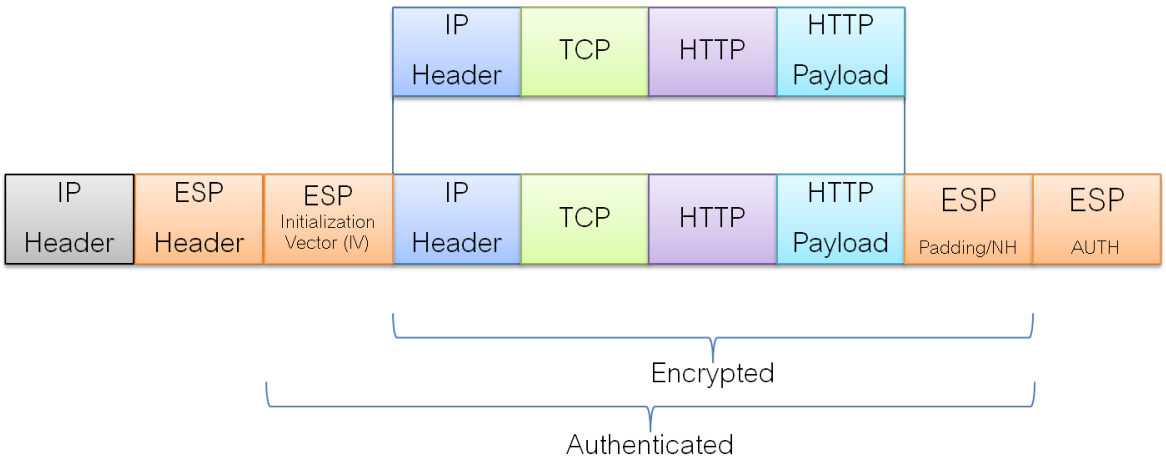


Figure 5.7: HTTP audio streaming protected with IPsec

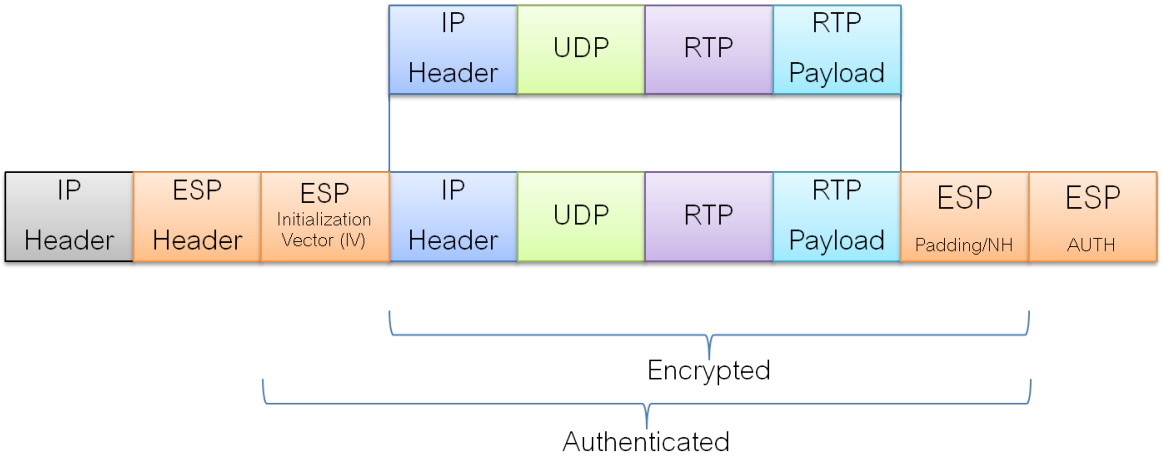


Figure 5.8: RTSP audio streaming protected with IPsec

authentication data, and thus may add 0-255 bytes of padding. The ESP Pad Length and Next Header fields must be right aligned in order to ensure that the ESP Auth (a.k.a. ICV, *Integrity Check Value*) is aligned on a 4-bytes boundary. The ESP Pad length is 1-byte length, as well as the following ESP NH field (Next Header). ESP NH is used to identify the protocol of the payload. Finally, in order to ensure integrity of the data flow, we use SHA-1 as integrity check algorithm, and the ESP Authentication Data Field contains a variable-length field computed over the ESP header, ESP IV, payload, and ESP trailer fields (Padding, Pad Length and Next Header).

5.4.2.2 Encryption algorithms: AES128-CBC vs. AES128-CTR

In terms of encryption, we have tested two different encryption algorithms: AES128-CBC and AES128-CTR. Both encryption algorithms use AES128 as block cipher. However, different modes of operation are tested: Cipher-Block Chaining (CBC) and CounTeR (CTR).

1. CBC: the mode of operation CBC is illustrated in figure 5.9. For encryption, the whole clear message is separated in blocks of 16 octets. Whenever the last block length is smaller than 16 octets, some padding is added in order to complete the block. Then, the first block of plaintext is XORed (represented with a circular plus symbol) with a Initialization Vector (IV). Notice that an IV is a block of bits used with the purpose of randomizing the encryption of some data (i.e. even if the same text is encrypted several times, the result is different each time). Then, the result of the XOR function is encrypted using AES algorithm and a key of 128bits. Finally, this ciphertext is used as IV for the next block of data and the process is repeated. Hence, this mode of operation does not allow parallelized encryption. Nevertheless, the decryption process allows parallelization. The blocks of ciphertexts are directly decrypted by the AES decryption algorithm together with its corresponding key. Only the first block of cipher text is subject to XOR processing with an IV. The rest of data blocks are XORed with the precedent ciphertext block (allowing parallelization of the decryption process).

2. CTR: the mode of operation CTR is shown in figure 5.10. This mode of operation allows using the same cipher block for encryption and decryption, it means that the decryption key scheduling do not need to be implemented, making it simpler than other mode of operations. For encryption, the key and a counter block of data are used as input of the block cipher. This generates a random keystream which is XORed with the clear text in order to obtain the ciphertext. The encryption process can be parallelized as there are no chaining operations, allowing independent cipher block inputs. For decryption, the same counter and keys are inputs of the block cipher and the ciphertext is XORed with this keystream, which results in the plaintext again. However, during decryption, if the

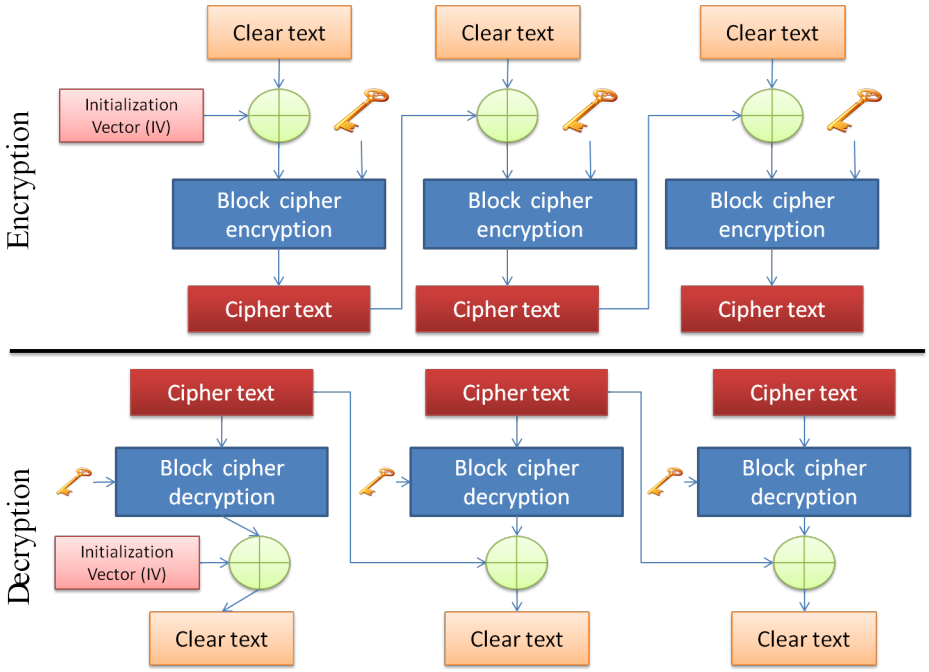


Figure 5.9: AES128-CBC encryption

keystream is longer than the clear text or ciphertext, the additional keystream bits are discarded. That is why, AES-CTR does not require padding completion to a multiple of the block size (see [42] and [43] for details).

5.4.2.3 Audio tools supporting several bit rates: 8kbps, 48Kbps or 96Kbps

We used two implementations in order to transmit and record the audio streaming from the server towards the EU during our tests: VLC and Arecord.

1. **VLC** stands for VideoLAN Client and server. It is an open source multimedia and streaming player. We have used this software in order to perform audio streaming in a client-server fashion. It supports various streaming protocols (refer to [44] for details). Depending on the test, parameters on the command line are modified in order to change the bit rate as well as the transmission protocol. We have tested three different bit rates: 8Kbps, 48Kbps and 96Kbps, as well as two different protocols: HTTP and RTSP (as explained in section 5.4.2.1). VLC allows to output any stream to a network or a file. Additionally, it allows processing the original streamed file in order to apply transcoding, filters, among other signal treatments. Appendix **??** shows a list of commands to perform audio streaming using different scenarios (i.e. protocols and bit rates). Finally, VLC also includes different modules that have different capabilities. The modules used for our tests are listed below:

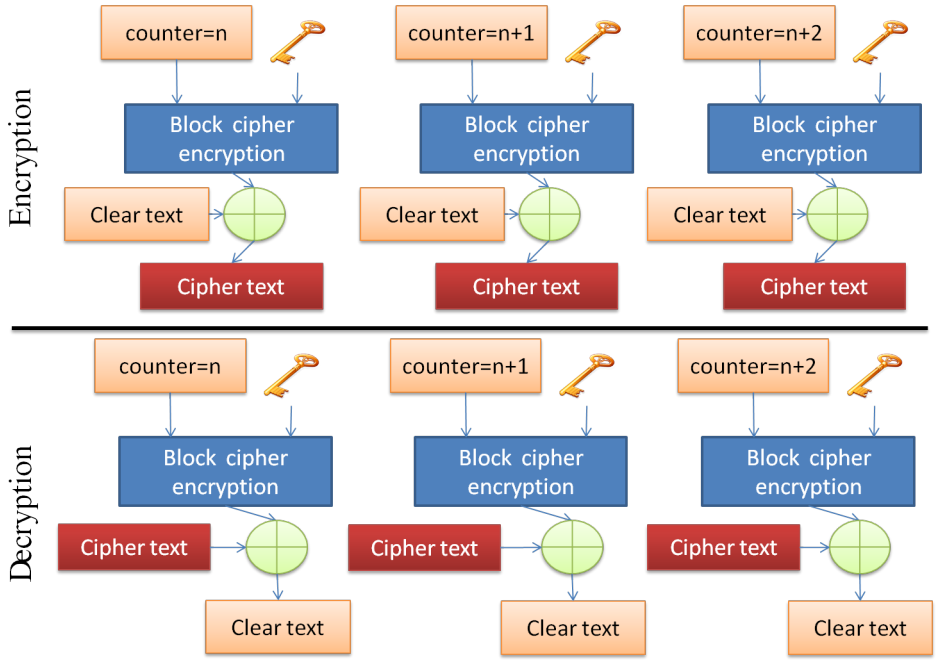


Figure 5.10: AES128-CTR encryption

(a) *Standard*: allows sending the stream to the network or saving it into a file. It has an option called '*access*' which specifies the protocol of transmission (e.g. RTSP or HTTP).

(b) *Transcode*: is used to decode and re-encode the streamed file using different bit rates (e.g. 8Kbps, 48Kbps and 96Kbps).

2. **Arecord** is a native Linux command for audio recording (see [45]). It supports a large set of soundcards and devices. The following command line captures the audio that is received during the streaming:

   arecord -f cd -d $vlc_timeout_duration /tmp/sound_card_record.wav

   The parameters of the command are explained below:

   -f, –format=FORMAT : format of the sample recorded. There exists a set of formats, however, we used a format shortcut "*-f cd*", which represents a 16 bit little endian, 44.1KHz and stereo capture. This is the equivalent of a CD-quality wave file.

   -d , –duration=N : interrupt recording after *N* seconds. Setting it to zero, means infinity. We set this value to a variable called $vlc_timeout_duration.

   File path : where the audio files is saved. In our case, */tmp/* directory. Then, at the end of the test, the captured audio file is transmitted to a server with the rest of test logs.

5.4.2.4 Software for measuring audio QoS: POLQA

In terms of quality of service measurements (QoS), we use *Perceptual Objective Listening Quality Assessment*, POLQA (see [17]). It is a relatively recent standard (2006-2011) for voice quality testing technology which is available under license. Orange, as a leading mobile operator in France, gets some licenses for using this software. Testing and measuring the impact of our developments over the QoS for some audio streaming, is a manner to evaluate the performance of our results.

5.5 Performance results

The results are represented in graphs with box-and-whiskers style (a.k.a. quartiles), as in section 4.7.1 (refer to section 3.4.3 for more details). Section 5.5.1 includes the results concerning the HA testbed considering different parameters (bit rates, frequency of interruption, etc.) whereas section 5.5.2 shows the results concerning the context management testbed. For each testbed, we analyze the network performances of our solution and then we concentrate on the QoS performances. We represented graphically all the results considering the parameters showed in table 5.1, and for each combination of parameters 25 measures were done in order to polt statistical results as quartiles.

5.5.1 Results for High Availability

This section exposes the results obtained during the performance tests concerning our High Availability solution. Section 5.5.1.1 shows the network performace and section 5.5.1.2 shows the impacts on QoS for each considered parameter.

5.5.1.1 Network performances:

Our first result shows how a VPN session is impacted while the HA scenario takes place. Figure 5.11 and 5.12 illustrate the switching time for all different scenarios and parameters presented in section 5.4.2 during an audio streaming of $8sec$. The term *switching time* is considered as the network time that a VPN session is interrupted while being transferred from SG1 to SG2 (or vice-versa). During a HA event (e.g. from SG1 to SG2), we calculate the *switching time* by subtracting the time when the first packet arrives from SG2 by the time when the last packet comes from SG1. This is done through the analysis of network captures during tests.

Figure 5.11 represents the case where the switching frequency (a.k.a. frequency of interruption) is $f=1/8s$. It illustrates the time to switch from SG1 to SG2 according to the bit rate used: 8Kbps, 48Kbps or 96Kbps. The left side of the graph represents HTTP and

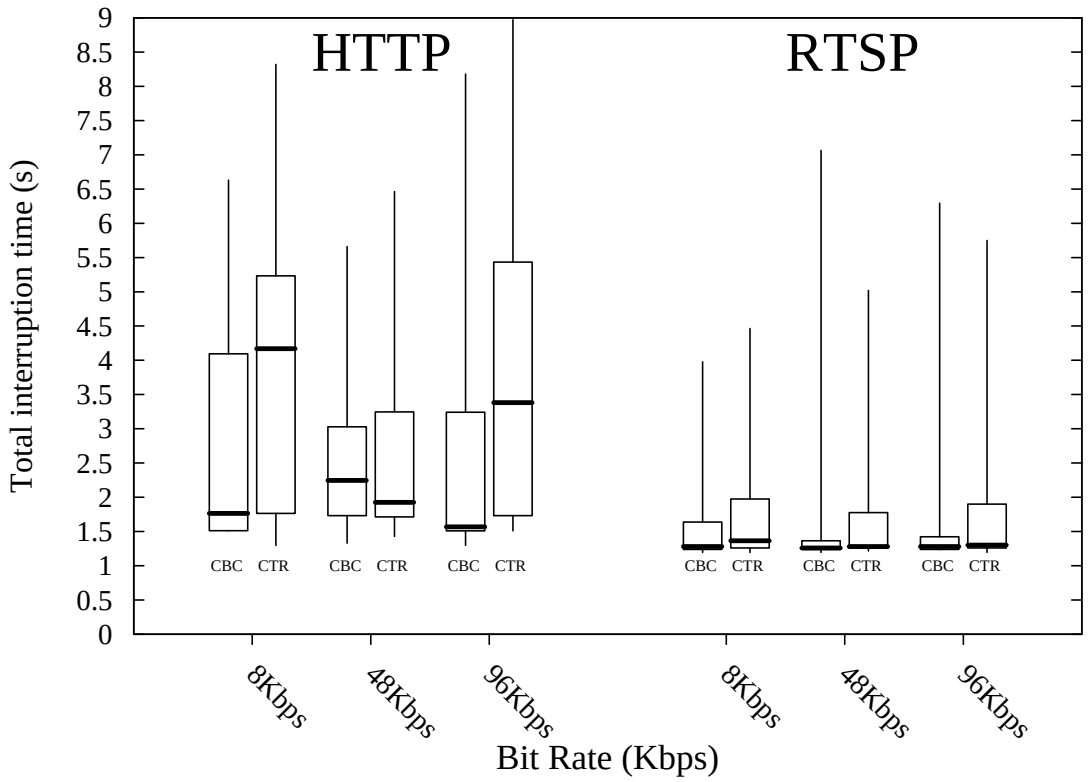
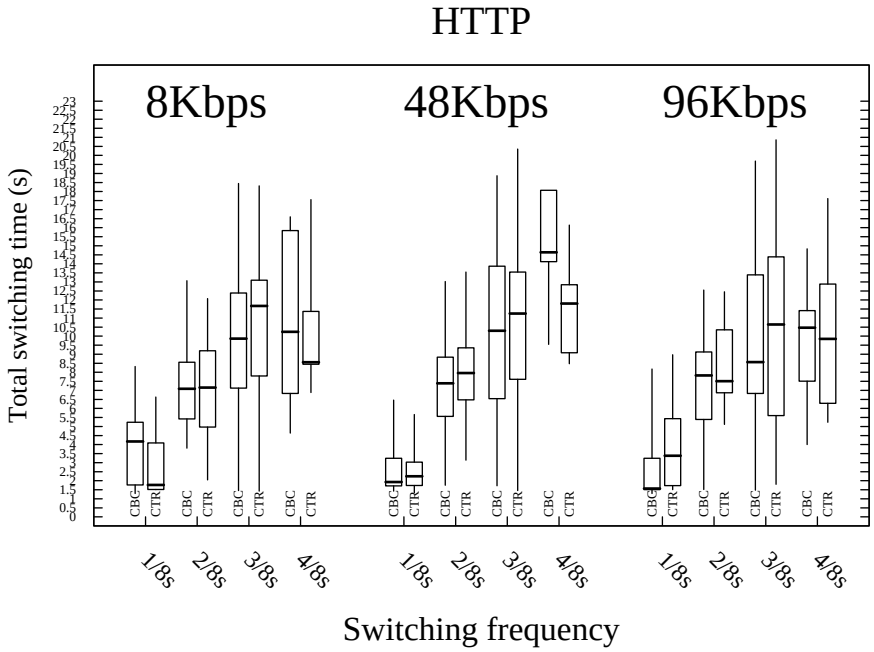


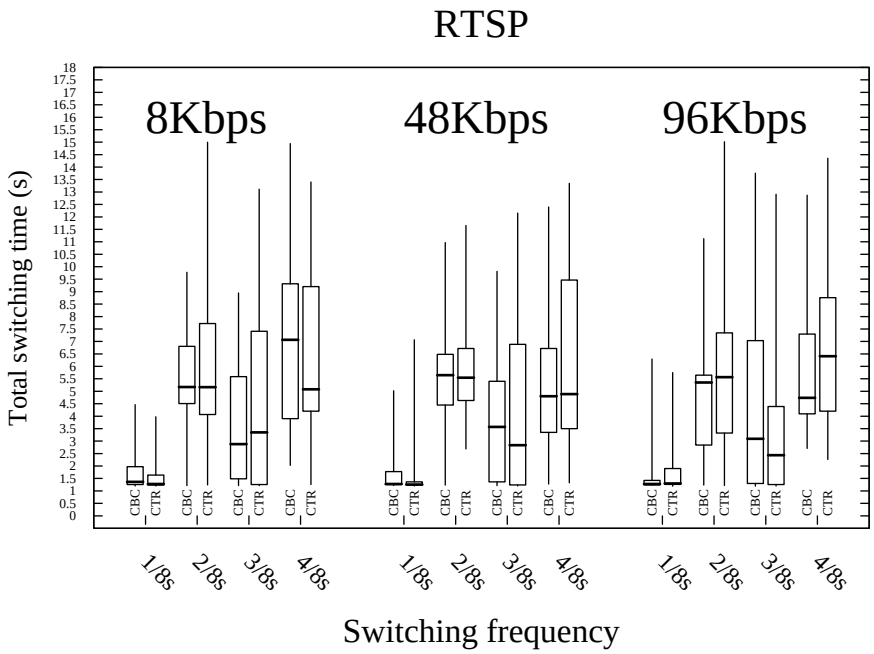
Figure 5.11: Total switching time for one interruption ($1/8s$). High Availability - Audio source file duration: 8 seconds

the right side corresponds to RTSP. The HTTP-CBC medians remain around $2sec$, but the quartiles are distant. This proves that HTTP is considerably impacted under an HA scenario. For HTTP-CTR, whose quartile are also spaced, the medians starts from $2.5sec$ (for 48Kbps) until $4.2sec$ (for 8kbps), being HTTP-CTR-8Kbps the worst median performance and less stable result of all cases for a switching frequency f=$1/8s$. On the other hand, in most scenarios, CBC had a better performance than CTR (unless for HTTP-48kbps only). Considering RTSP, the switching time is notably stable around $1.5sec$, and the quartiles are narrower, evidencing stability of the measure. This means that regardless of the encryption algorithm and application/transport protocol used, this scenarios are fairly stable. Notice that we do not pretend to compare HTTP against RTSP as application protocols. The objective is to explain the difference of using whether HTTP or RTSP and to measure how our developments of HA impacts them in different circumstances. Actually, we decided to use HTTP and RTSP, because HTTP relies on TCP whereas RTSP relies on UDP protocol, allowing to study the impact over connection oriented and non-connection oriented protocols. However, figure 5.11 illustrates that is better to implement RTSP than HTTP while streaming under a HA environment.

Figure 5.12 illustrates the switching time caused by different switching frequencies from $1/8s$ to $4/8s$ (refer to 5.4.1 for details). In order to recreate a more realistic environment, the



(a) Protocol: HTTP



(b) Protocol: RTSP

Figure 5.12: Switching time for several interruption frequencies ($^1/_8s$,$^2/_8s$,$^3/_8s$ and $^4/_8s$). High Availability - Audio source file $8sec$

interruptions are randomly generated on time so that failover events never take place at the same time. Figure 5.12a illustrates the total switching time when using HTTP whereas figure 5.12b represents RTSP.

Both figures 5.12a and 5.12b show an unstable response facing the different switching frequencies. Even though the quartiles are spaced, the total switching time increases linearly when streaming over HTTP. In the case of RTSP, the total switching time remains stable during frequency $^1/_{8s}$, but quartiles become more spread when the frequency of interruption increases. We noticed through network captures that the HTTP audio streaming packet sizes are considerably bigger than those packets of an RTSP audio streaming. This explains why HTTP audio streaming gets more impacted in terms of network performance, usually taking more time to overcome a failure event. As HTTP is based on TCP, this connection-oriented protocol ensures reliability, ordering, and error-checking of delivered stream of octets. This situations leads to a considerable overhead and higher impact during interruptions, as more information must to be retransmitted. Thus, loosing some information when using HTTP streaming might represent more troubles than streaming with RTSP. Due to the connection-less oriented behavior of RTSP (based on UDP), those packets that are lost are not sent again, which represents lower overhead.

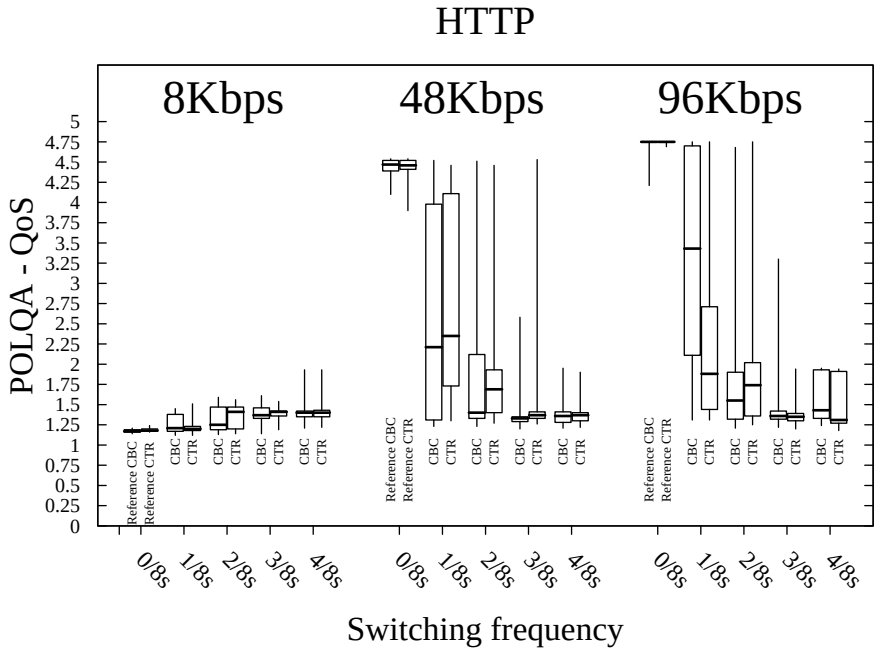
5.5.1.2 Quality of Service:

Figures 5.13 and 5.14 show all the results in terms of QoS. POLQA's scores from zero (low quality audio) to five (high quality audio).

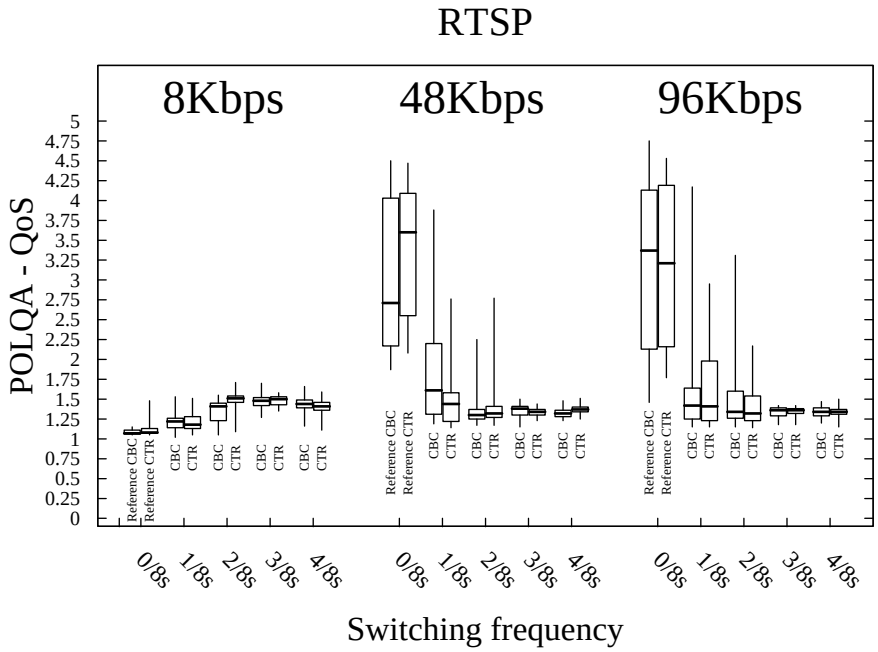
Figures 5.13a and 5.13b represent the results of the QoS for different switching frequencies, which are represented as $^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$. We also added quartiles for frequency zero ($^0/_{8s}$), which represents the reference QoS for each case. This allows comparing the QoS deterioration as long as we increase the switching frequency. We observe that as soon as the switching frequency increments, both 48Kbps and 96Kbps (HTTP and RTSP) downgrade the QoS. However, when using 8Kbps bit rate, the QoS smoothly improves as the switching frequencies increase.

A bit rate of 8Kbps is considered as a very bad quality by POLQA, and the note always remain below of $^{1.25}/_5$. During interruptions, POLQA gives a slightly better QoS than the reference QoS, meaning that the interruptions are actually similar to an audio with compression of 8Kbps. On the other hand, 48Kbps and 96Kbps decrease in terms of QoS regardless of the encryption algorithm while the switching frequency increases.

Finally, figure 5.14a illustrates the loss of QoS when the switching frequency is $^1/_{8s}$. This figure is also partially represented in figure 5.13, but this time we show the results according to the bit rate (instead of switching frequencies). Also, we added an additional quartile called Reference QoS. In fact, the Reference QoS consist of measuring the quality of an audio streaming while no interruption takes place. This is how we can actually measure the impact of the interruption



(a) Protocol: HTTP. Architecture: High Availability



(b) Protocol: RTSP. Architecture: High Availability

Figure 5.13: QoS for several interruption frequencies ($^1/_{8}s$,$^2/_{8}s$,$^3/_{8}s$ and $^4/_{8}s$). High Availability - Audio source file duration: 8 seconds

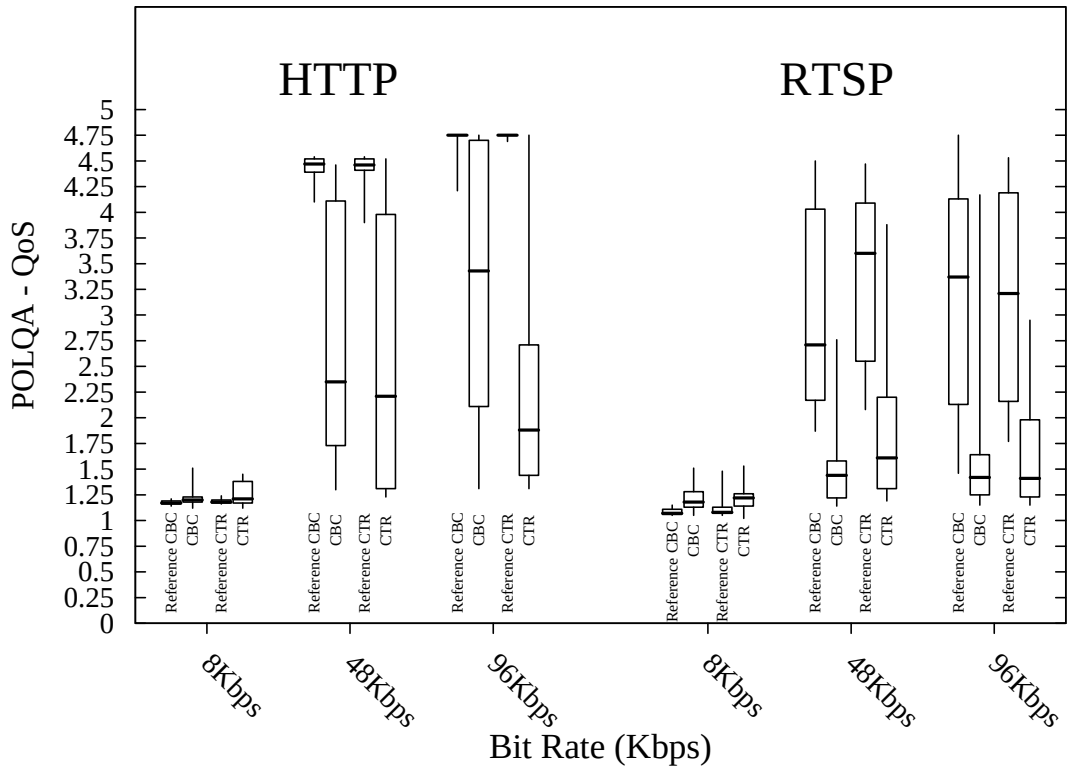


Figure 5.14: QoS measurements for one interruption frequency ($1/8s$). High Availability - Audio source file: 8 seconds

by comparing it with different switching frequencies ($1/8s$,$2/8s$,$3/8s$ and $4/8s$).

By comparing the switching frequency f=$1/8s$ with the reference QoS, we obtained the following main results: the QoS for HTTP-48Kbps is degraded by around 50%, HTTP-96Kbps-CBC by 35% and HTTP-96Kbps-CTR by 58%. The 8Kbps results are particularly different. The POLQA's algorithm gives a very bad note to any transmission made with 8Kbps. Audio streaming at this rate does not accomplish a good level of quality, thus obtaining an average note of $1.23/5$. Even for interruption where switching frequency is $1/8s$, POLQA gives a better note as the reference QoS. This means that interruptions are better noted than the audio transmission itself due to its bad quality. For further details in how POLQA algorithms objectively evaluates audio quality, refer to [46]. Table B.2 summarizes the results of QoS as long as the switching frequencies increase for all the cases.

5.5.2 Results for context management

This sections exposes the results obtained during the performance tests for our context management scenario. Section 5.5.2.1 shows the network performance and section 5.5.2.2 presents the impacts on QoS for each considered parameter.

High Availability	Reference QoS ($^0/_{8s}$)	$^1/_{8s}$	$^2/_{8s}$	$^3/_{8s}$	$^4/_{8s}$
HTTP-8Kbps-CBC	100%	+3%	+6%	+17%	+17%
HTTP-8Kbps-CTR	100%	0%	+19%	+19%	+19%
HTTP-48Kbps-CBC	100%	-50.5%	-68%	-70%	-69%
HTTP-48Kbps-CTR	100%	-52%	-62%	-69%	-69%
HTTP-96Kbps-CBC	100%	-27%	-67%	-71%	-69%
HTTP-96Kbps-CTR	100%	-60%	-63%	-71%	-72%
RTSP-8Kbps-CBC	100%	+14%	+31%	+38%	+34%
RTSP-8Kbps-CTR	100%	+9%	+39%	+39%	+30%
RTSP-48Kbps-CBC	100%	-40%	-52%	-49%	-51%
RTSP-48Kbps-CTR	100%	-60%	-63%	-62%	-61%
RTSP-96Kbps-CBC	100%	-57%	-60%	-59%	-60%
RTSP-96Kbps-CTR	100%	-56%	-58%	-57%	-58%

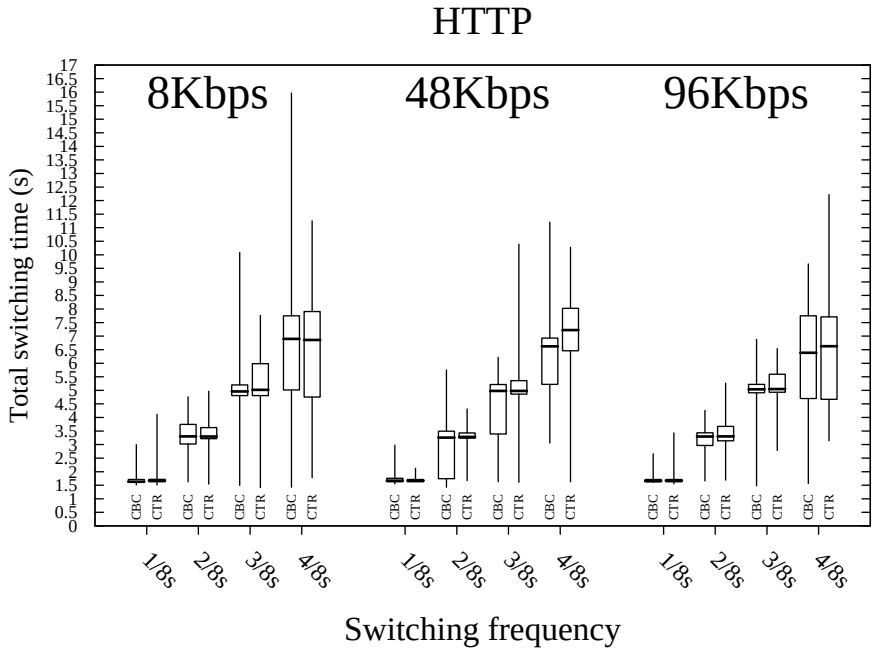
Table 5.2: Impact of the frequency of high availability events over QoS (considering the medians)

5.5.2.1 Network performances:

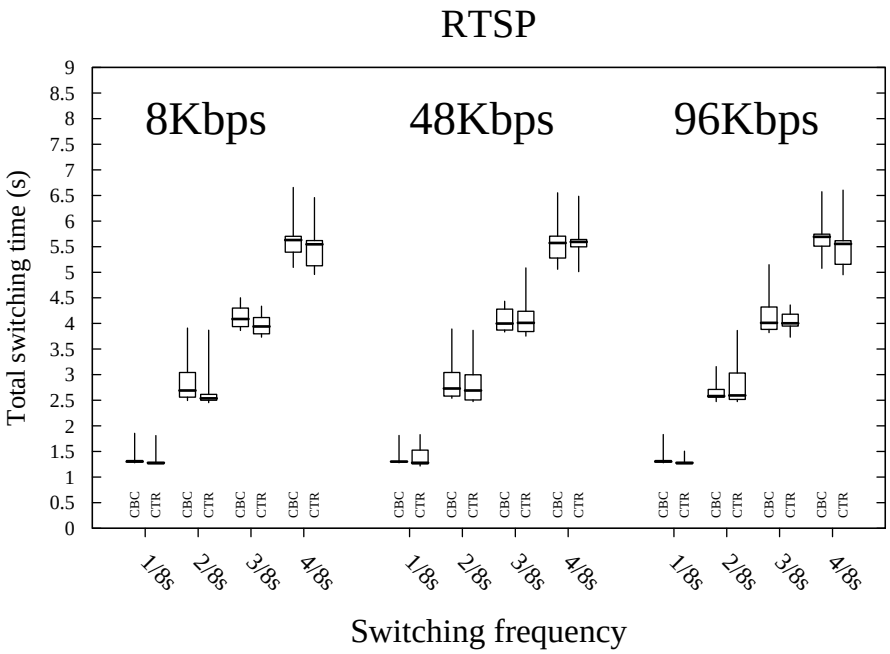
Figures 5.15 and 5.16 represent the results of an IKEv2/IPsec context transfer under the context management testbed, considering all the parameters and scenarios described in section 5.4.2. Notice that measurements take place during a streaming transmission of an audio source file with a duration of $8sec$.

Figure 5.15 illustrates the total switching time for different switching frequency cases. As for the HA testbed, the total switching time is calculated as the accumulated time when switching from SG1 to SG2 and vice-versa: $^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$. Figure 5.15a and 5.15b show a linear tendency upwards. Quartiles for the HTTP scenario are more spaced than those of RTSP, showing that RTSP is more stable in terms of network performance than HTTP during interruptions. Regardless of which encryption algorithm is implemented, CBC and CTR algorithms show similar performances. Only HTTP-48Kbps indicates a little advantage of CBC compared to CTR, but in general terms, both algorithm perform similarly. As our testbed includes a 2-core CPU's, this is the main reason why it does not make a huge influence in terms of total switching time. For RTSP protocol (fig 5.15b), CTR showed a 3% to 5% better performance than CBC.

Figure 5.16 shows the switching time in function of different bit rates, considering a switching frequency of $^1/_{8s}$. For both protocols, the median remains very stable. We notice that RTSP performs better than HTTP. In fact, HTTP takes in average 25% more time to switch from SG1 to SG2 than RTSP. Although the quartiles are narrow, sometimes HTTP takes $3sec$ to $3.5sec$ to perform a context management event (i.e. HTTP-CBC-8Kbps and HTTP-CTR-96Kbps). Concerning the cryptographic algorithms, CBC and CTR obtained very similar results. We observe that regardless of the bit rate used, this has no impact over the type of encryption.



(a) Protocol: HTTP



(b) Protocol: RTSP

Figure 5.15: Total switching time for several interruption frequencies ($^1/_{8}s$,$^2/_{8}s$,$^3/_{8}s$ and $^4/_{8}s$). Context management - Audio source file: 8 seconds

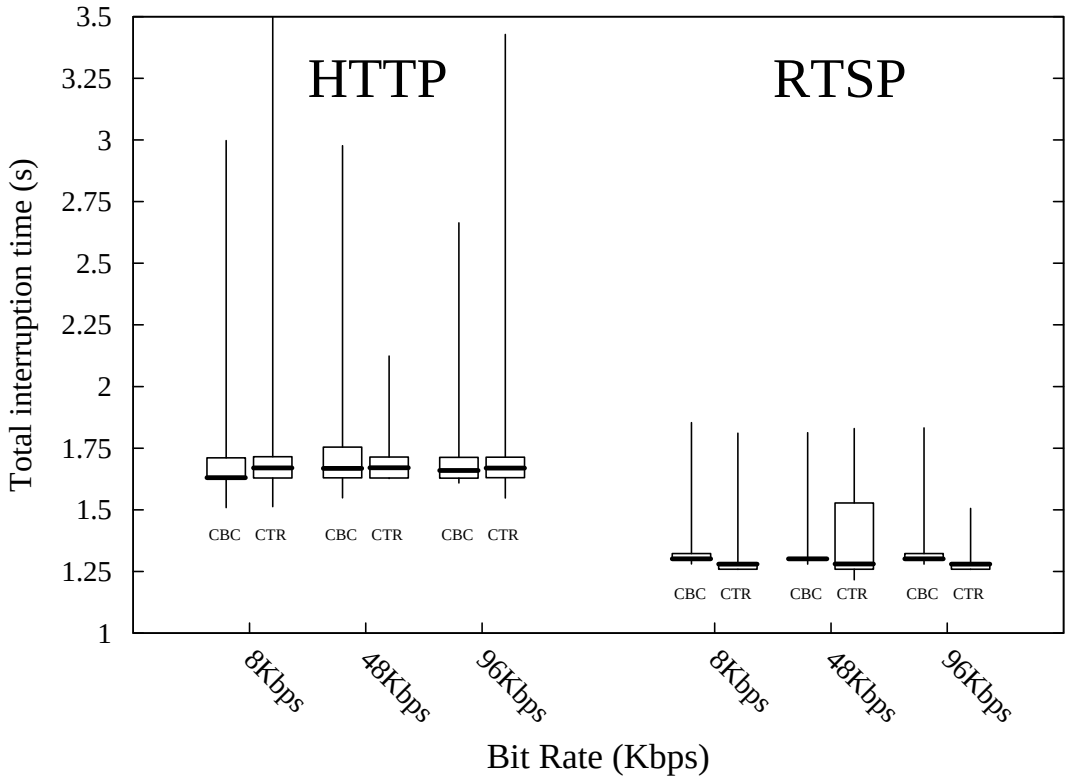


Figure 5.16: Switching time for one interruption ($^1/_{8s}$) - Context management - Audio source file: 8 seconds

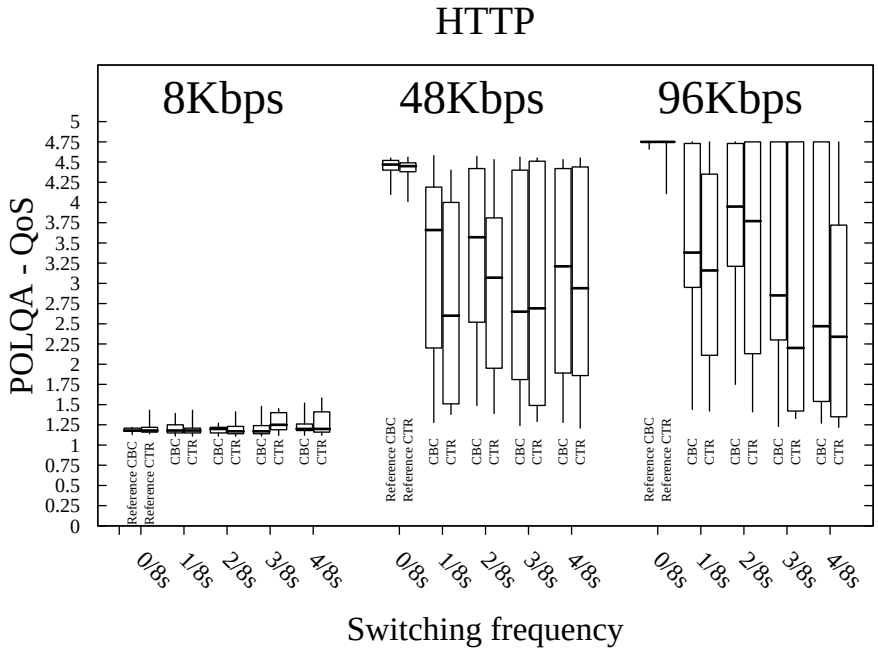
5.5.2.2 Quality of Service:

Figures 5.17 and 5.18 illustrate results in terms of QoS. We consider the same scenarios and parameters as for the network performance results, so we can compare them.

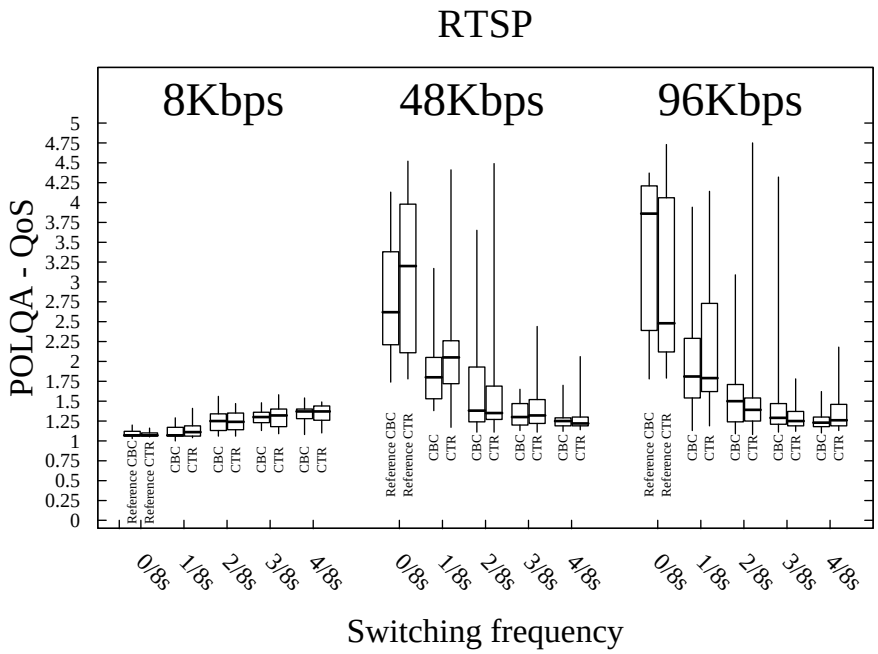
Figure 5.17 shows the QoS for several switching frequencies ($^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$). As for the HA testbed (refer to 5.5.1.2), we also added quartiles for frequency zero ($^0/_{8s}$). This represents the reference QoS for each considered scenario. This allows comparing the QoS deterioration as long as we increase the switching frequency.

Figure 5.17a represents the variation of the QoS while streaming under HTTP. 8Kbps obtains poor marks as in HA testbed ($^{1.25}/_5$). In terms of QoS, 8Kbps is strongly discouraged to use. Actually, by performing audio streaming at 8Kbps we might gain in terms of bandwidth but loose in terms of quality. On the other hand, HTTP-48Kbps and HTTP-96Kbps both maintain acceptable good notes as long as the switching frequency increases. Even though the QoS is logically downgraded while increasing the amount of interruptions, HTTP maintains a fair QoS. Note that HTTP uses TCP as transport protocol, which ensures lost packets to be resent, maintaining a good behavior in terms of QoS. However this may represent high bandwidth consumption as well as longer delays when listening to the audio streaming. For example, in terms of network performance, HTTP shows higher interruption levels when increasing the switching frequencies than RTSP (see fig 5.15). Figure 5.17b represents the QoS for several switching frequencies with RTSP, showing that it is clearly more impacted in terms of QoS than HTTP. Although RTSP showed good results in terms of network performance by switching from SG1 to SG2 in $1.3sec$, this is not the case in terms of QoS. In fact, as RTSP is based on UDP, connectionless-oriented protocols do not resend lost packets, introducing higher impacts on the QoS of the streaming. When using RTSP, the impact over the QoS for $^1/_{8s}$,$^2/_{8s}$,$^3/_{8s}$ and $^4/_{8s}$ switching frequencies are (in average): 36,5%, 50%, 55,75% and 57,5% correspondingly.

Finally, figure 5.18 shows the QoS evaluation for different bit rates with a switching frequency f=$^1/_{8s}$. We observe that RTSP is more impacted than HTTP. HTTP's QoS decreases by 18% and 28% (CBC-48Kbps and CBC 96Kbps) while RTSP's QoS decreases by 31% and 53% (CBC-48Kbps and CBC 96Kbps). However, in some cases RTSP has less impact on the QoS than HTTP in percentage (i.e. -41% using HTTP-48Kbps-CTR and -35% using RTSP-48Kbps-CTR), but in fact, the reference QoS is also staled since the beginning (i.e. $^{4.5}/_5$ for HTTP-48Kbps-CTR against $^{3.25}/_5$ for RTSP-48Kbps-CTR). This means that, in some cases, the HTTP streaming might be a little bit more impacted than RTSP streaming, but the QoS remains the best when using HTTP. Table B.3 summarizes the evaluation of the QoS as long as the switching frequencies increase on a context management scenario.



(a) Protocol: HTTP



(b) Protocol: RTSP

Figure 5.17: QoS for several interruption frequencies ($^1/_8s$,$^2/_8s$,$^3/_8s$ and $^4/_8s$) - Context management - Audio source file: 8 seconds

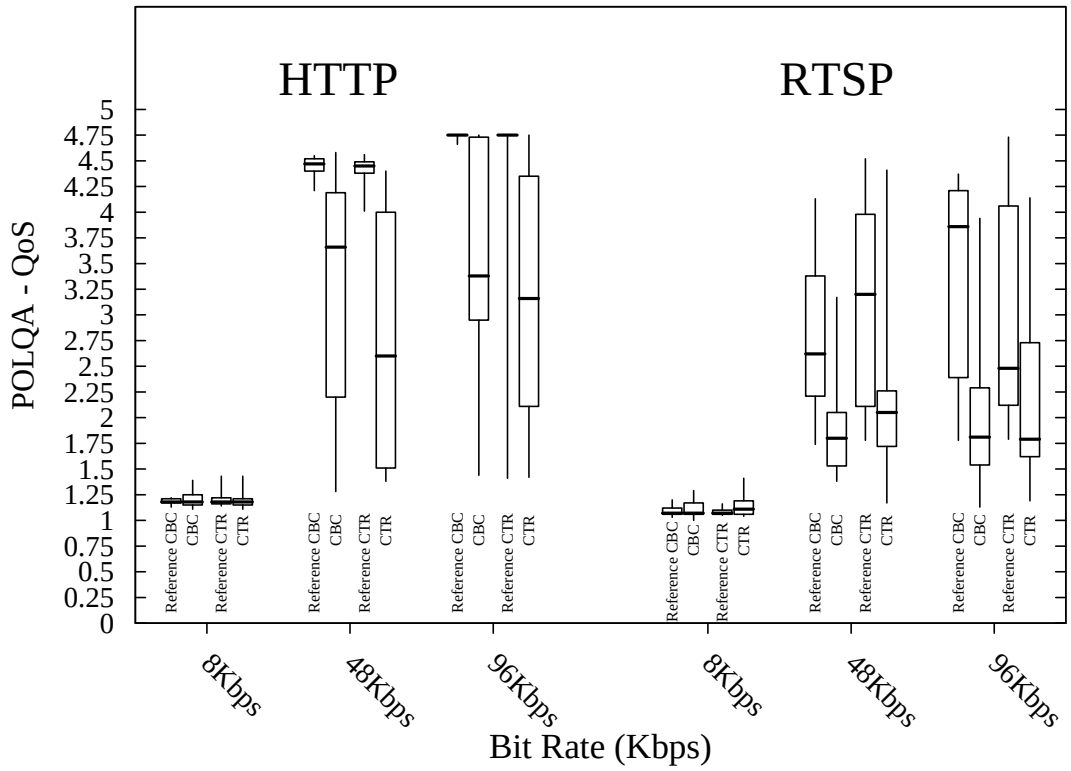


Figure 5.18: QoS measurements for one interruption frequency ($^1/_{8s}$). Context Management - Audio source file: 8 seconds

Context Management	Reference QoS ($^0/_{8s}$)	$^1/_{8s}$	$^2/_{8s}$	$^3/_{8s}$	$^4/_{8s}$
HTTP-8Kbps-CBC	100%	0%	+1%	0%	+1%
HTTP-8Kbps-CTR	100%	0%	0%	+5%	+1%
HTTP-48Kbps-CBC	100%	-18%	-20%	-40%	-28%
HTTP-48Kbps-CTR	100%	-41%	-31%	-39%	-33%
HTTP-96Kbps-CBC	100%	-28%	-16%	-40%	-48%
HTTP-96Kbps-CTR	100%	-33%	-20%	-53%	-50%
RTSP-8Kbps-CBC	100%	0%	+16%	+21%	+28%
RTSP-8Kbps-CTR	100%	+3%	+15%	+23%	+28%
RTSP-48Kbps-CBC	100%	-31%	-41%	-50%	-52%
RTSP-48Kbps-CTR	100%	-35%	-57%	-58%	-61%
RTSP-96Kbps-CBC	100%	-53%	-61%	-66%	-68%
RTSP-96Kbps-CTR	100%	-27%	-43%	-49%	-49%

Table 5.3: Impact of the frequency of context management events over QoS (considering the medians)

5.6 Conclusions

Throughout this chapter we described the IKEv2/IPsec context transfer under a Mono-LAN environment, where a cluster offering a VPN service is configured with a common IP address. Contrary to chapter 4, this time our scenarios included the IKEv2/IPsec context transfer between physically different SGs. We defined two new components: the heartbeat and the sync. These components allows to synchronize every tunnel that is negotiated with any of the cluster members, and the heartbeat regularly checks liveness of the nodes.

By introducing the High Availability architecture, we added the capability to successfully overcome a failover event thanks to the constant replication of the IPsec and IKEv2 data. We found out that an ISPs may be interested in deploying this architecture to ensure reliability of the VPN service. On the other hand, by increasing the availability of the system, the End-User experience and the QoS is improved when avoiding re-authentication when moving from one SG to another. Also, the IKEv2/IPsec context management under a Mono-LAN environment represents advantages for the VPN service. Indeed, an ISP wanting to balance the load among different cluster members might use this feature to dynamically move an IPsec session from one SG to other. This improves the performance of the VPN service and avoids overloaded gateways within the cluster.

We tested two different scenarios: High Availability and Context Management. We also consider several parameters throughout the tests: encryption methods, bit rate transmissions as well as connection and non-connection oriented transport protocols. We found out that the context management scenario performed better than HA. Actually, even if both scenarios contemplate the transfer of an IKEv2/IPsec context, they perform differently and might be implemented in different circumstances. The heartbeat component of the HA scenario adds automatism during failover occurrences, however the transfer of the IKEv2/IPsec context only happens once the heartbeat gives the order to do so. Instead, under a Context Management scenario, an ISP is able to move an IPsec session whenever needed, and thus improving the performance and End-User experience. This is the main reason why the context management scenario performed better than HA.

Our results are presented in terms of network behavior and QoS. During the HA tests, we noticed that the network behavior of non-connection oriented protocols (i.e. UDP) are less impacted by a failover event than a connection oriented protocol (i.e. TCP). In fact, due to the nature of TCP which mandates to acknowledge transmitted data, this generates additional delays to recover from failover events. However, from a QoS point of view, TCP obtained better results than UDP. Indeed, with UDP, those packets that are lost during failover events are not recovered, thus degrading the QoS. Considering the context management, we noticed that connection oriented protocols are less impacted compared to the scenario where HA features

are enabled. This is mainly due to the time that the heartbeat takes to perform its task. Actually, during a context management, there is no heartbeat involved during the transfer of the IKEv2/IPsec context, and thus, the SG taking responsibility of a connection does not need to wait the liveness check to install the new IPsec session. When comparing HTTP (TCP) with RTSP (UDP), we noticed that RTSP has a better network behavior than HTTP. However, in terms of QoS, HTTP obtains (once again) better results than RTSP.

We also measure the Mean Opinion Score (MOS), a measure of voice quality, with POLQA. From the three different bit rates used, 92kbps consumes more capacity, but delivers an average MOS of 4.75; 48kbps delivers a MOS slightly better than 4.3 and 8kbps delivers a MOS of 1.25. Thus, an enterprise must decide whether MOS values deliver acceptable quality or whether they would prefer to trade some bandwidth for higher quality.

Future work includes interaction of IPsec facing a context transfer between SGs with different IP addresses. Even though we improved the VPN service by creating the cluster of nodes with a single IP address, an ISP may also be interested in moving an IPsec session between distant SGs with different IP addresses. We believe that this may be advantageous to ensure the continuity of a VPN service for a mobile user that is probably changing its attachment point. This is explained in details in chapter 6.

Part III

IPsec/IKEv2 Context Transfer for Multi-LAN architectures

Chapter 6

Multi-LAN IKEv2/IPsec Context Transfer

This chapter introduces the IKEv2/IPsec context transfer considering SGs with different IP addresses. This is what we call a **Multi-LAN** context transfer. Actually when a SG transfers an IKEv2/IPsec context to another SG, all the security parameters and cryptographic material are also transfered as fast as possible in order to avoid incoherent or unsynchronized states (especially IPsec counters). Throughout this chapter, we introduce a mechanism based on MOBIKE extension of IKEv2 (discussed in section 2.6) allowing to change the attachment point of a peer during an active IPsec session. This includes the theoretical process to achieve such mechanism as well as we provide some developments recommendations. The practical tests and developments for this scenario are left as future work.

The chapter is organized as follows: section 6.1 describes the motivation of performing Multi-LAN context transfers, then, section 6.2 positions our work compared to related topics concerning the transfer of an IPsec context. Following section 6.2.3 focuses in explaining MOBIKE extension and how it can be implemented to perform IKEv2/IPsec context transfers (subsection 6.4). Finally we present our conclusions in section 6.5.

6.1 Motivations

An IKEv2/IPsec context transfer should be performed smoothly with respect to End-Users. Our goal is to avoid the number of exchanges between the access network devices (SGs) and the End-User during a context transfer. However, when the IP address associated to an IPsec VPN tunnel changes, it is mandatory to update the corresponding IPsec and IKEv2 databases on both extremities. This introduces some delays that might impact the EU experience during an active IKEv2/IPsec session. We propose a solution by performing a **GET** action from the SG sending

the context, and then performing a **PUT** action at the SG receiving the IKEv2/IPsec context. However, in contrast to precedent chapters, an additional INFORMATIONAL IKEv2 message must be exchanged between the EU and the SG in order to inform that the IP address changed. This impacts the EU experience as it is involved during the transfer between SGs. Nevertheless, MOBIKE is an ideal extension of IKEv2 allowing to perform our Multi-LAN context transfer, as it avoids re-authentication from scratch, reducing delays in the establishment of a new tunnel towards a new IP address.

Even though the mobility feature of MOBIKE was initially designed as to be applied from the client side (which is supposed to be a mobile node), our solution can use this exchange at the SG's side, and update the IP address where a peer is attached to. However, this would introduce some changes to the source code of the implementation of the protocol itself, as we will be using a mobility protocol to actually perform an IKEv2/IPsec context transfer. Section 6.2.3 explains the details of our proposed solution.

6.2 Position of our Work

This section includes the description of some related work, considering our Multi-LAN scenario. First we address a generic mechanism that manages to exchange some defined contexts (in our case IPsec). Then, we position our work considering a mechanism called REDIRECT, allowing to perform redirection of peers toward different SGs, and finally we finish by describing MOBIKE.

6.2.1 Context Transfer Protocol (CXTP)

The IETF's experimental protocol, CXTP [7], enables transferring contexts for various services (e.g. QoS, Security, Authentication, etc.) between different SGs. A Context Transfer (CT) can be either *Mobile Controlled* (initiated by End-Users) or *Network Controlled* (initiated by the newly active SG **nSG** or the previously active SG **pSG**). Once the CT is activated, it may happen that one of the end-points already knows towards which destination the context will be transferred. This case is known as **predictive** mode, and it is the most favorable scenario in terms of packet loss and performance, mostly because the context transfer takes place before the handover is done. On the other hand, if the context transfer occurs abruptly, it is known as **reactive** mode. Obviously this mode is less beneficial for an IKEv2/IPsec connection in terms of packet loss and performance. We should consider using this protocol in order to transmit the IKEv2/IPsec related information.

6.2.2 RFC5685 - Redirect Mechanism

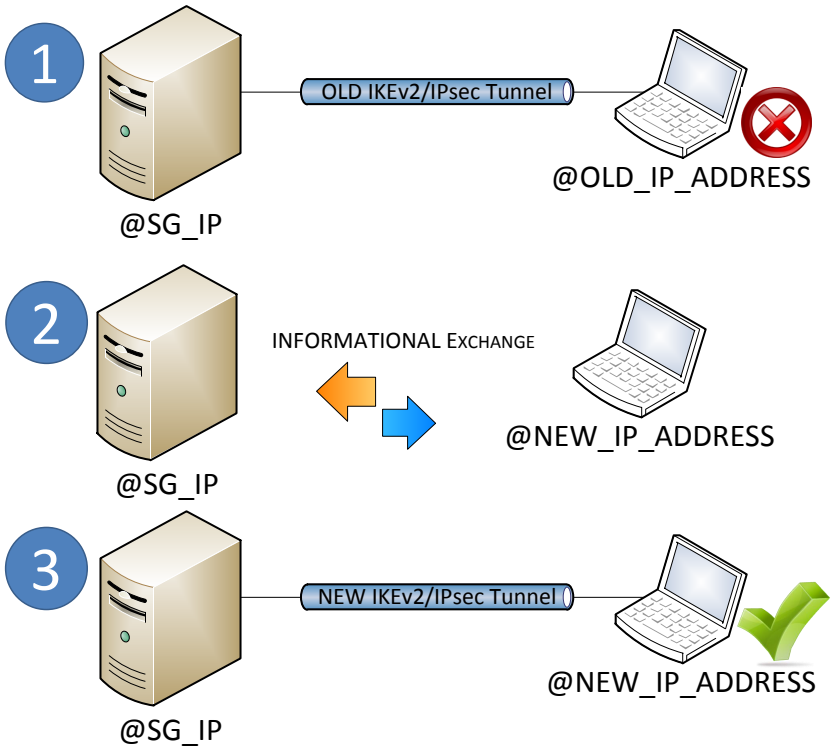
The RFC5685 describes a method (see 6.2 and [4]) that allows to redirect an IKEv2/IPsec session from a previously active SG towards a newly active SG. This mechanism does not pre-authenticate an EU against the newly active SG, actually the EU needs to renegotiate the cryptographic information from scratch. Our proposal would be to combine this REDIRECT mechanism with the transfer of the IKEv2/IPsec context, therefore the EU would not need to re-authenticate its session again. However, this includes changes to the standard. New payloads must be added in order to carry out smooth context transfer during an active session. Therefore, if for some reason an active SG is being shut down (e.g. failure, maintenance, overload, etc.), it would be possible to REDIRECT a peer to another SG. The REDIRECT action could be performed during IKE_INIT, IKE_AUTH (once authentication have been verified) or during an active IKEv2/IPsec session by using an INFORMATIONAL exchange using a new notify payload. On the other hand, there is no software implementation of this standard at the moment, which is a barrier to make us believe this is the right protocol to be implemented for our Multi-LAN scenario in the near future.

6.2.3 MOBIKE - IKEv2 Mobility and Multihoming Protocol

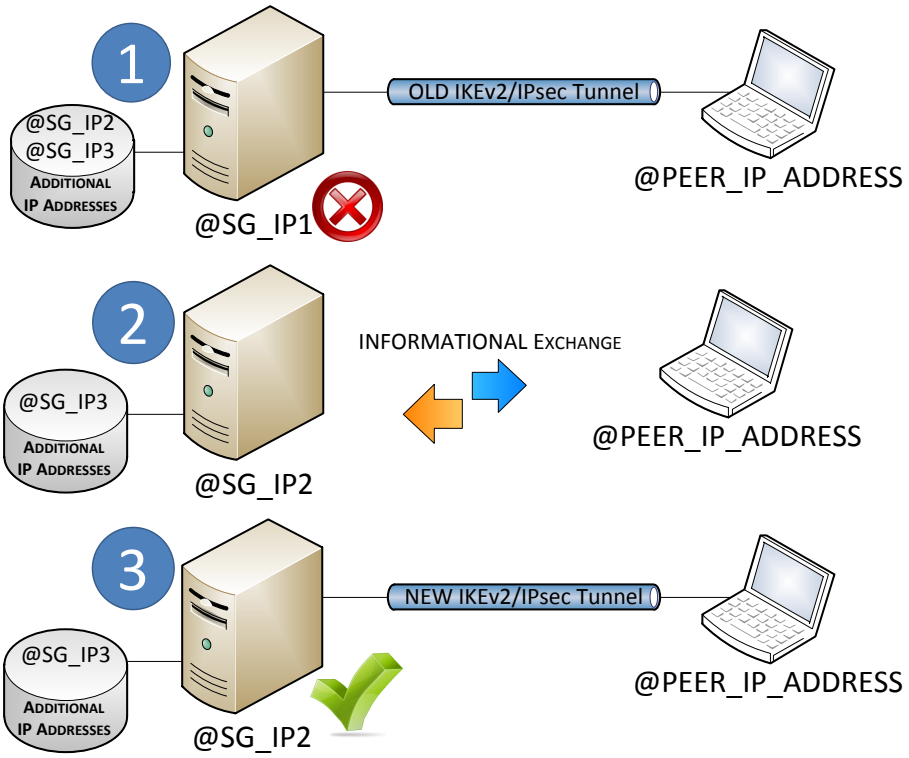
As described in section 2.6, MOBIKE is an extension of IKEv2 allowing to perform mobility and multihoming features during and active VPN session. Initially, IKEv2 is the protocol used to perform mutual authentication during establishment of the tunnel as well as to perform maintenance of the IPsec_SAs involved. The IKE_SAs and IPsec_SAs are created between the IP addresses that are used during the IKEv2 initial exchanges. These IP addresses are used as the outer IP header of the IKEv2 further exchanges and the protected traffic as well. However, there are some use cases where these IP addresses might change. For example, a peer that changes its point of attachment to the network might change its IP address too, loosing communication with the SG and the initial tunnel that was already established. Another example would be a device that has multiple interfaces with multiple IP addresses wishing to switch its traffic from one interface to another.

What happens when one of the IP addresses of the tunnel change?

There are two possible solutions when these IP addresses change. First, one of the extremities of the tunnel can re-initiate a new negotiation of IKE_SAs and IPsec_SAs from scratch. However, this is not the most beneficial situation for an ongoing session. Creating a new VPN tunnel involves expensive calculation and further round-trips that will downgrade the EU experience. For example, authentication exchanges often ask a password to End-Users (e.g. a pin code), introducing additional delays that may negatively impact the VPN session and the service above the IP layer. Second, if we manage to use the already established IKEv2/IPsec in order to inform



(a) Peer's mobility with MOBIKE as in RFC4555



(b) Multihoming feature with MOBIKE as in RFC4555

Figure 6.1: Mobility and Multihoming performed with MOBIKE extension

the other peer that the IP address has changed, then it allows re-using the already established VPN session in order to update the IP address on both extremities of the tunnel. This avoids round-trips as well as additional calculation while creating a new IKEv2/IPsec context.

Figure 6.1 represents the two main scenarios where MOBIKE is used for. The first scenario, shown in figure 6.1a, represents a peer that changes its IP address during an active IPsec session from @OLD_IP_ADDRESS to @NEW_IP_ADDRESS. At Step 1, for some reason @OLD_IP_ADDRESS is not reachable anymore. At step 2, the peer acquires a new address @NEW_IP_ADDRESS and launches an INFORMATIONAL IKEv2 exchange informing the other peer that its IP address has changed. Then, at step 3 the IKEv2/IPsec databases are updated and the tunnel is maintained without creating a new one from scratch. The second scenario (figure 6.1b) contemplates a SG with multiple interfaces and different IP addresses. This is called the multihoming feature. At step 1, the initial @SG_IP1 is not reachable anymore. At step 2, the SG uses one of its additional IP addresses in order to inform the other peer that it has changed its IP address (from @SG_IP1 to @SG_IP2). At step 3, the IKEv2/IPsec session is maintained by updating the IP address on both extremities of the tunnel. The communication goes on and the multihoming feature ensures continuity of the IKEv2/IPsec session.

6.3 Multi-LAN Scenario

The scenario we address with the Multi-LAN context transfer is shown in figure B.6. It illustrates a set of n SGs offering a VPN service to an EU. Indeed, the EU reaches a server (e.g. some service proposed by the provider) by establishing a secure connection with one of SGs, authenticating and encrypting the communication between them. At some point of the VPN session, it is preferable for the operator to change the attachment point of the EU towards some other SG. The operator has then two choices: either it deletes its actual session and let the EU establish a new VPN tunnel towards a new SG, or it activates a modified MOBIKE exchange in order to accomplish the migration of SG. Of course, it is not suitable for an operator to break off the VPN session, as the EU is negatively impacted. Section 6.4 introduces the usage of MOBIKE as a solution to perform IKEv2/IPsec context transfers in order to improve the EU experience while maintaining its VPN session alive.

6.4 Proposed solution based on MOBIKE extension

MOBIKE has been implemented by different VPN solutions (software and hardware). It is also implemented within *strongSwan*, the main open source project for Linux. This makes MOBIKE the ideal candidate to be used in order to perform smooth IKEv2/IPsec context transfer between SGs owning different IP addresses. Our goal is to use the MOBIKE extension in order to change

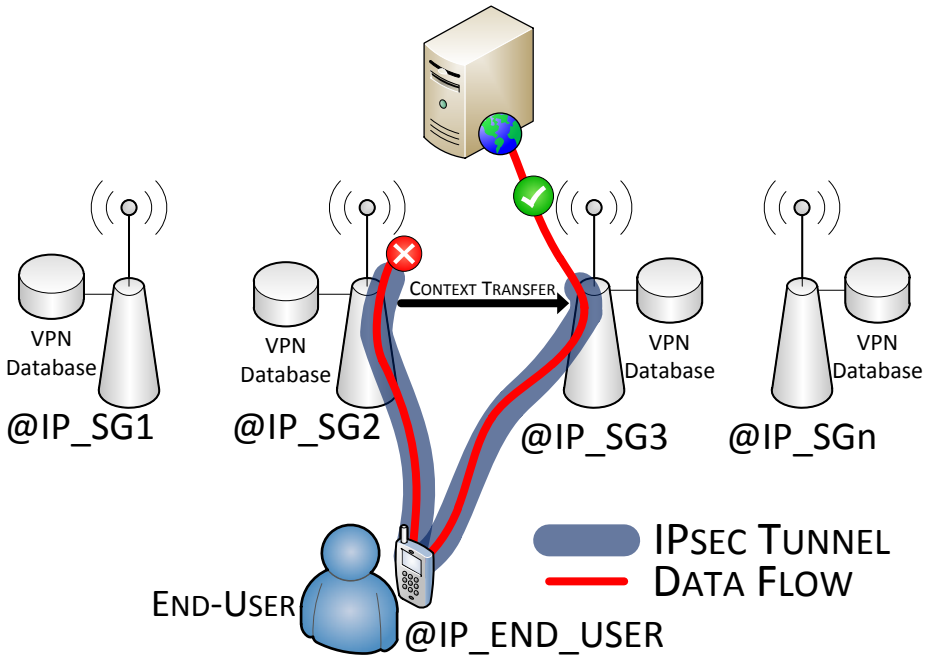


Figure 6.2: Scenario of a Multi-LAN IKEv2/IPsec context transfer

the IP address of the SG. Even though this case is similar to the original multihoming feature of MOBIKE showed in figure 6.1b, the main difference is that the new IP address of the SG, is owned by a physically different SG. Instead of forwarding packets to an ADDITIONAL_IP_ADDRESS within the same SG, the traffic is now forwarded to an IP address belonging to a new SG. Nevertheless, the new SG must also receive the IKEv2/IPsec parameters of the tunnel in order to ensure the continuity of the VPN service, otherwise it would not be possible to decrypt traffic coming from the EU.

Figure B.5 illustrates a schema allowing to use MOBIKE with the transfer of the IKEv2/IPsec context. At step 1, before mobility occurs, the old SG transmits the IKEv2/IPsec context to the new SG. Once the context is received, during step 2, the new SG initiates a MOBIKE exchange in order to inform the other peer that the VPN tunnel has changed its IP address to @SG_IP2. At step 3, the IKEv2/IPsec databases are up to date concerning the new IP address of the SG side. Now, the continuity of the VPN service is ensured.

What are the impacts of using Transport Mode and Tunnel Mode?

As discussed in section 2.3, transport mode and tunnel mode represent two different uses cases of IPsec. Considering the transfer of an IKEv2/IPsec context, both scenario might have different impact in terms of IP mobility. When tunnel mode is used, the outer IP header of the IPsec-protected traffic is changed, whereas the inner IP address of the tunnel stay unchanged. This allows applications not to be interrupted at the IP layer. Nevertheless, when transport mode is used, the new header of the IPsec-protected traffic generates an interruption at the IP layer, and the applications might consider to manage IP mobility.

6.4.1 Implementation considerations

During the transfer of the IKEv2/IPsec between SGs with different IP addresses, there are some factors that might be evaluated and considered:

1. SPI collision: during the transfer of the IKEv2/IPsec session, it could be possible that a collision of SPI (Security Parameter Index) occurs. To solve this, the implementation may consider an additional IKEv2 exchange in order to negotiate a new SPI. The probability that a collision occurs is very low, but if this ever happens, the IKE_SA and corresponding CHILD_SAs would tear down.

2. Trigger: there should be a trigger to carry out the transfer of an IKEv2/IPsec context. For example, one reason could be a mobile node changing its attachment point due to low signal, and thus it may be beneficial to transfer the tunnel IKEv2/IPsec from one SG to another. Some other reasons could be: a SG reaching its maximum capacity and needing

to avoid getting overloaded, an EU wishing to reduce costs of its connectivity, better QoS, among others.

3. Counters: as already discussed in section 4.4, during the IKEv2/IPsec context transfer it might be advantageous to increase the *Sequence Number* value in order to avoid stale counters. On the other hand, once the context transfer takes place, the IKEv2 exchanges MUST be stopped on the old SG and should be processed uniquely by the new SG; otherwise this would cause stale *Message ID* values. Nevertheless, the implementation can also consider an additional IKEv2 exchange as in RFC6311 in order to negotiate the values of the counters (refer to [15]).

4. Refreshing keys: the implementation may enforce its security by re-keying the IKEv2/IPsec cryptographic material once the context transfer is finished.

5. Context Definition: as discussed in section 6.2.1, transferring the IKEv2/IPsec context can be done by implementing the Context Transfer Protocol (CXTP). However, we need to define the framework allowing to transfer all the parameters to ensure the continuity of the VPN service.

6.5 Conclusions

Throughout this chapter, we present how to proceed when performing a context transfer of the IKEv2/IPsec session based on a mobility extension, MOBIKE. This solution may have positive impacts concerning the EU experience whenever a VPN session is transferred between different devices. Actually, as we have discussed in previous chapters, an IPsec session is initially intended to be maintained within the same device that established the connection. However, from a ISP point of view, there are advantages to transfer an IPsec session between different devices.

We defined the Multi-LAN scenario as in figure B.6. We also described the exchanges during the IKEv2/IPsec context transfer in figure B.5. We expect to reduce the amount of calculation and time that the ISP spend during session establishments. Indeed, the energy consumption of the ISP would be reduced as long as the IKE_SAs and IPsec_SAs are kept during the transfer of an IKEv2/IPsec context. This improves the actual behavior of the network in terms of energy and traffic management.

We detected the difference between transport mode and tunnel mode of IPsec. In tunnel mode, the context transfer is virtually transparent to EUs. Indeed, the IPsec traffic (i.e. ESP traffic), is actually tunneled within a payload containing a new outer IP header (the inner IP header remains the same). However, when IPsec is used in transport mode, the upper layers implementations must deal with IP mobility as well, as the IP header of the IPsec traffic changes

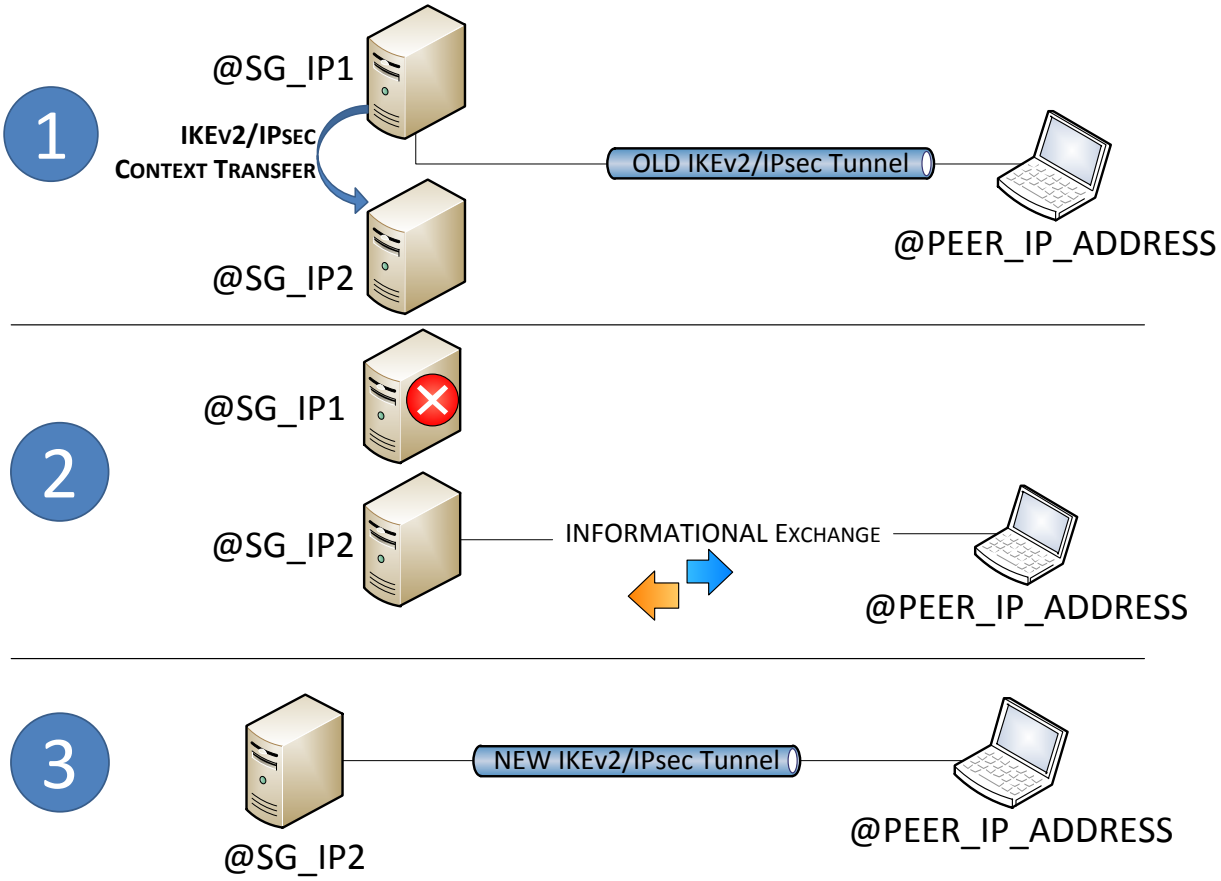


Figure 6.3: Multi-LAN IKEv2/IPsec context transfer

too. We have already done some investigation in terms of mobility for IPsec transport mode (see [32, 37, 38]), where we designed a protocol called MOBIKE-X.

Future work includes the implementation of the solution as described in section 6.4. A new research contract extension has been approved mutually between Orange (France Telecom) and the University of Paris 6 in order to continue this research topic.

Chapter 7

Conclusions and Future Work

This thesis is focused on enhancing and ensuring the continuity of service of IKEv2/IPsec based communications, mainly implemented as VPN services.

The first part of our investigation focused on how to protect IP networks with IPsec. We also introduced the concept of using IKEv2 as the perfect complement to IPsec in order to perform authentication of parties and subsequent management of IPsec connections. This allows operators to offer a reliable and secure environment over untrusted network. Also, by providing an introduction of some IKEv2 extensions like MOBIKE or REDIRECT, we pretend to present a framework for seamless context transfer in order to dynamically manage an IKEv2/IPsec session.

Following investigations lead us to develop two new features concerning IKEv2/IPsec VPNs: GET and PUT functions. Our main motivation was to achieve dynamic management of a VPN session whenever desired. We found out that extracting all the information concerning an IKEv2/IPsec session would take around $15ms$, and re-installing it would take $22ms$ approximately. Our implementations were done in *StrongSwan*.

The second part of the thesis is dedicated to the Mono-LAN environment. We studied the implementation of ClusterIP, a mechanism allowing to create a cluster of SGs under a single virtual IP address. Results in chapter 4 showed that an active SG spends 5% to 8% of CPU more than a passive SG when clustering IPsec tunnels with ClusterIP. We also observed that there is an additional load when using the HA-plugin of strongSwan among the cluster members. Further investigations, includes merging ClusterIP features with our own functions (GET and PUT). We created a framework of seamless IKEv2/IPsec context transfer. This drove us to the definition of two new architectures offering High Availability and Context Management capabilities based in our GET and PUT implementations. We presented our results in terms of network behavior and quality of service. During HA tests, we realized that the network behavior of non-connection oriented protocols (i.e. UDP) are less impacted by a failover than

a connection-oriented protocol (i.e. TCP). Concerning context management results, we noticed that connection oriented protocols are less impacted compared to the scenario where HA features are activated. Throughout the second part of the thesis, we notice that these implementations allow providing load balancing as well as load sharing properties among different cluster members without having a dedicated device. However, we also found out that these solutions are more specific for locally established clusters, where SGs are placed physically close while remaining within a same network segment.

Finally, the third part of the thesis is dedicated to Multi-LAN environments. We defined an architecture for transferring IKEv2/IPsec contexts between SGs owning different IP addresses. Although both tunnel and transport modes involve some IKEv2 signalization when changing its attachment point (i.e. INFORMATIONAL exchange), they are impacted differently. In tunnel mode, when the IP address associated to the end-point of the tunnel changes, this interruption is transparent for upper layers applications. Even though the outer IP address of a tunnel changes, the inner IP address of the tunnel remains the same. On the other hand, for transport mode communications, there is no inner IP address. Applications would have to deal with IP mobility whenever its IP address associatied to an IKE_SA and IPsec_SA changes. Our efforts concentrate on allowing an operator to move an EU from one SG to another by using minimal signalization, less consumption and less calculation associated to the construction of cryptographic material.

Global Evolution and Future Work

Since most of the terminals are now equipped with alternative access network interfaces (e.g. WiFi), we believe that operators have to focus on how they can be used to get access to services. On the other hand, one of the potential evolutions to accommodate traffic from mobile access networks is to offload the Radio Access Network (RAN) traffic towards complementary access networks such as WiFi, WIMAX, femtocells, etc. These networks are called to maintain the same level of security and perform as well as in RANs. Additionally, non mobile access networks, like WiFi, are widespread as wireless extensions of broadband access networks, mainly in indoor spaces where cellular technologies do not perform well.

ISPs are interested in offering new capabilities to its VPN services. Even if today's multimedia contents are mainly offered by Web services (HTTP streaming, VoIP, etc.), the access to the network itself is still managed by ISPs. Operators shall be able to easily adapt to new business opportunities. For example, architecture choices can make it easier for Orange to offer NaaS (Network as Service) services in the future, and this requires implementing VPNs. Hence, enhancing the reliability, availability and quality of such services is a matter of interest.

As discussed throughout this investigation, mobile data traffic is expected to grow annually by 60% to 70%. Even though the traffic is increasing more and more, there have also been generalization of flat rate billing plans, higher availability of attractive contents and increasing

availability of high consumption devices (e.g. laptops, smartphones, tablets, etc.). All this context creates great pressure over operators for them to provide the best possible performance over their networks.

Our own developments in this area lead us to manage a VPN tunnel. Allowing an operator to dynamically handle the EUs that use its VPN services, definitively have a beneficial impact for the EU experience. However, more work remains in order to make it fully work within operational networks.

First, even if our testbeds concerning Mono-LAN environments considered the transfer of the IKEv2/IPsec context between two different SGs, we believe that it is important to design a framework in order to decide responsibility of incoming request when the cluster is composed of n nodes. For HA scenarios, we addressed active/passive scenarios, but for load sharing or load balancing scenarios, it would be important to define an algorithm that decides how to spread the load among the different cluster members.

Concerning Multi-LAN environments, we presented a solution based on MOBIKE allowing to perform context transfer. This solution addresses the context of an IKEv2/IPsec context between nodes with different IP address. As the period of the thesis came to its end, this implementation is indeed part of the future work of our investigation. However, a very motivating project emerged in order to continue our research in this area. Once the developments of Multi-LAN are done, we pretend to give the community a mechanism that ameliorates the behavior of the network. Actually, future networks tend to be more heterogeneous, and this would be a mechanism that allows to switch a VPN session between different SGs. Also, we pretend to reduce the delays of reestablishing a new VPN tunnel towards a new SG, instead of transferring this data between the access routers themselves.

It might be important to concentrate efforts in defining some sort of local intelligence in order to manage IKEv2/IPsec sessions. A sort of application capable to externalize the decision of managing sessions. For example, a use case would be a Connection Manager deciding whether to perform handover, mobility or multihoming features, and to dynamically manage security aspects. As IKEv2 is the protocol allowing to negotiate modifications to IPsec, the connection manager would be able to issue orders in order to customize an active IKEv2/IPsec session.

Nowadays networks are composed of several access technologies, where each one defines its own security standards, introducing heterogeneity to the network. This situation has the drawback that it is complex to maintain the same security level among different access network technologies, which may include different devices (constructors) as well. However, RANs have now migrated to IP based communications. This allows us to propose IPsec as a security mechanism all over the network. Our interest is to continue our investigations in order to introduce more sophisticated mechanisms that allow maintaining the same security

level regardless of the access technology. We pretend to propose an IETF draft in order to push the standardization towards the IPsec context transfer.

Appendices

Appendix A

VLC Streaming Commands

A.1 Commands

This section provides additional information about technical commands introduced in VLC in order to achieve audio streaming.

Action	Command Line	Explanation
HTTP Streaming (server side)	*vlc input_stream –sout '#transcodeacodec=mp4a, ab=bit_rate #standard{access=http, mux=ts, dst=example.org:port_number}'*	Set the server side in order to stream the audio file using HTTP protocol. *input_stream* is the audio file streamed, mux=ts sets the multiplexing to MPEG2/TS and bit_rate determines 8Kbps, 48Kbps or 96Kbps.
HTTP Streaming (client side)	*vlc http://example.org:port_number*	Request from the client an audio streaming through HTTP protocol (based on TCP).
RTSP Streaming (server side)	*vlc –sout ' #transcode{acodec = mp4a, ab = bit_rate} : rtpdst = client_address, iport = client_port, sdp = rtsp://example.org:server_port/ file_name '*	Set the client side to receive the audio streaming based on UDP. The bit_rate parameter defines the transcoding of the audio streaming (8Kbps, 48Kbps or 96Kbps).
RTSP Streaming (client side)	*vlc rtsp://example.org:port/file_name*	Set the client to receive the streaming using RTSP protocol (based on UDP).

Figure A.1: Commands for VLC server and client

Appendix B

Résumé étendu en Français

B.1 Résumé

En 2012, le trafic mobile mondial représentait 70% de plus qu'en 2011. L'arrivée de la technologie 4G a multiplié par 19 le volume de trafic non 4G, et en 2013 le nombre de mobiles connectés à l'Internet a dépassé le nombre d'êtres humains sur la planète. Les fournisseurs d'accès Internet (FAI) subissent une forte pression, car ils ont pour obligations d'assurer à leurs clients l'accès au réseau et le maintien de la qualité de service. À court/moyen terme, les opérateurs doivent délester une partie de leur trafic sur des réseaux d'accès alternatifs afin de maintenir les mêmes caractéristiques de performances. Ainsi, pour désengorger les réseaux d'accès radio (RAN), le trafic des clients peut être préférentiellement pris en charge par d'autres réseaux d'accès disponibles. Notons cependant que les réseaux d'accès sans fil offrent des niveaux de sécurité très différents. Pour les femtocells, WiFi ou WiMAX (parmi d'autres technologies sans fil), il doit être prévu des mécanismes permettant de sécuriser les communications.

Les opérateurs peuvent s'appuyer sur des protocoles (tels que IPsec) afin d'étendre un domaine de sécurité sur des réseaux non sécurisés. Cela introduit de nouveaux défis en termes de performances et de connectivité pour IPsec. Cette thèse se concentre sur l'étude des mécanismes permettant de garantir et améliorer les performances du protocole IPsec en termes de continuité de service.

La continuité de service, aussi connu comme résilience, devient cruciale lorsque le trafic mobile est dévié depuis un réseau d'accès RAN vers d'autres réseaux d'accès alternatifs. C'est pourquoi nous nous concentrons d'abord dans l'ensemble de protocoles assurant une communication IP : IKEv2 et IPsec. Ensuite, nous présentons une étude détaillée des paramètres nécessaires pour maintenir une session VPN, et nous démontrons qu'il est possible de gérer dynamiquement une session VPN entre différentes passerelles de sécurité. L'une des raisons qui justifient la gestion des sessions VPN est d'offrir de la haute disponibilité, le partage de charge ou

l'équilibrage de charge pour les connexions IPsec. Ces mécanismes ont pour finalité d'augmenter la continuité de service de sessions IPsec.

Certains nouveaux mécanismes ont été récemment mis en oeuvre pour assurer la haute disponibilité sur IPsec. Le projet open source VPN, StrongSwan, a mis en place un mécanisme appelé ClusterIP afin de créer un cluster de passerelles IPsec. Nous avons fusionné cette solution basée sur ClusterIP avec nos propres développements afin de définir deux architectures : une architecture permettant la Haute Disponibilité et une deuxième architecture présentant la gestion dynamique d'un contexte IPsec. Nous avons défini deux environnements : le Mono-LAN où un cluster de noeuds est configuré sous une même adresse IP unique, et le Multi-LAN où chaque passerelle de sécurité dispose d'une adresse IP différente.

Les mesures de performance tout au long de la thèse montrent que le transfert d'une session VPN entre différentes passerelles évite les délais supplémentaires liés à la ré-authentification et réduit la consommation CPU, ainsi que les calculs par le matériel cryptographique. D'un point de vue FAI, le transfert de contexte IPsec/IKEv2 pourrait être utilisé pour éviter la surcharge des passerelles, et permettre la redistribution de la charge, de meilleures performances du réseau ainsi que l'amélioration de la qualité de service. L'idée est de permettre à un utilisateur de profiter de la continuité d'un service tout en conservant le même niveau de sécurité que celui initialement proposé.

B.2 Objetif de la thèse

L'objectif de cette thèse est de permettre aux infrastructures VPN IPsec de fournir une haute qualité de service aux clients ainsi que d'assurer la continuité du service VPN. Plus précisément, nous voulons que les connexions VPN puissent résister aux échecs et défaillances. L'une des architectures abordées tout au long de cette thèse est l'haute disponibilité (High Availability, HA). Nous voulons aussi que les FAI puissent facilement gérer la charge due au trafic VPN de leurs clients (c'est ce que nous appelons gestion du trafic ou de gestion d'un contexte IPsec). Ainsi, lorsque une plate-forme VPN multi-nœuds est utilisé par le fournisseur d'accès Internet, celle-ci devrait être capable de transférer une connexion VPN d'un nœud à un autre, en introduisant le concept de transfert de contexte. Ainsi, quand une session VPN est transférée, cela signifie que tous les paramètres relatifs à cette session sont transférées également. En effet, il y a des scénarios où la possibilité de transférer une session VPN d'une SG à une autre aurait des effets positifs. Certains cas d'utilisation impliquent: l'équilibrage de la charge dû au trafic VPN entre les différents nœuds du cluster, le transfert entre différents réseaux d'accès ou l'amélioration de basculement entre clusters offrant la haute disponibilité (HA).

B.3 Description du Service VPN et ses challenges techniques

B.3.1 La Haute Disponibilité de VPNs IPsec

Un service VPN qui offre des fonctionnalités liées à la haute disponibilité permettra d'améliorer l'expérience de ses utilisateurs finaux en cas de panne au cours d'une session VPN active. En effet, entre les défis techniques afin d'offrir la haute disponibilité nous avons: la détection d'un SG qui ne fonctionne plus, ainsi que le fait de changer de passerelle tout en gardant la session VPN préalablement établie.

Lorsqu'un EU établie une VPN avec une SG, certains éléments cryptographiques ainsi que les clés sont dérivées entre eux. Quand la SG tombe en panne, nous ne voulons pas que l'utilisateur renégocie tous ces paramètres avec une autre SG qui est censé d'assumer la responsabilité de la connexion VPN. Ainsi, le matériel cryptographique et les clés associées doivent être partagés entre la SG défaillante et la SG de secours. L'ensemble de paramètres sont appelés sous le nom du Contexte IPsec/IKEv2 et sont expliqués en détail ultérieurement.

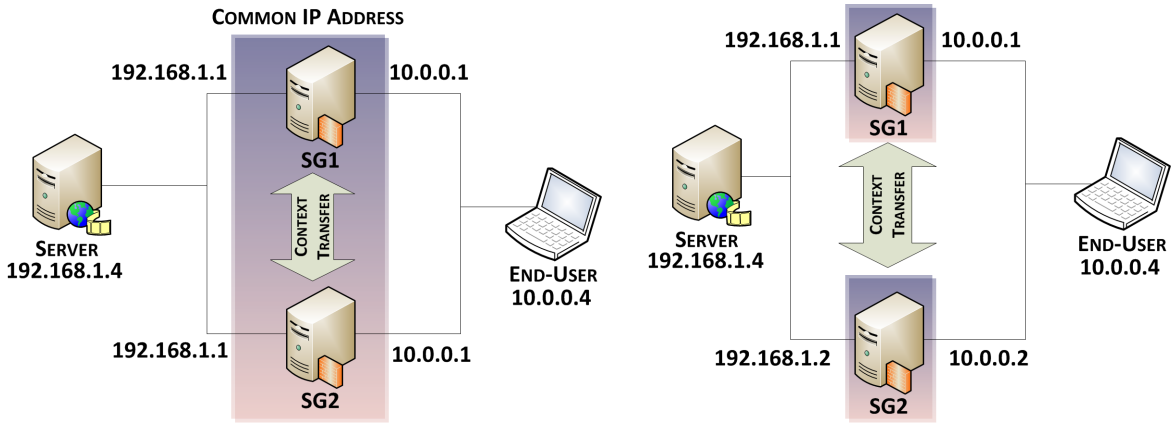
Nous allons considérer les architectures suivantes:

- **Mono LAN:** Cette architecture se compose d'un ensemble de SGs lesquelles possèdent la même adresse IP. Comme représenté dans la figure 1.4a, deux SGs partagent la même adresse IP. Toutefois, les deux SGs ne peuvent pas posséder la même adresse IP au même temps, sauf si un mécanisme spécifique est utilisé. Dans ce cas, l'utilisateur final n'aurait pas besoin de mettre à jour son adresse IP lors d'un changement de passerelle (par exemple depuis SG1 vers SG2), et le contexte IPsec/IKEv2 devrait également être transférée entre les deux SGs.

- **Multi LAN:**cette architecture se compose d'un ensemble de SGs possédant des adresses IP différentes. La figure 1.4a représente une architecture multi-LAN, où la SG1 et SG2 ont des adresses IP différentes. Dans ce cas, quand un transfert de contexte a lieu, l'utilisateur est impacté, car il doit mettre à jour l'adresse IP de destination de son tunnel VPN (l'une des extrémités du tunnel change). Le client peut faire face à ce problème en effectuant des échanges IKEv2 spécifiques tels que: MOBIKE ou REDIRECT.

La synchronisation des compteurs IKEv2/IPsec constitue le plus grand défi au moment de surmonter une défaillance dans une passerelle de sécurité. Au même temps, ces compteurs sont aussi un obstacle à l'heure d'offrir des fonctionnalités du type *haute disponibilité*. Ces compteurs contrôlent chaque paquet IP entrant/sortant d'un nœud protégé par IPsec. Ceux-ci préviennent le fait de rejouer des paquets non autorisés, notamment du trafic IPsec précédemment traité. Ils doivent rester synchronisées sur les deux extrémités du tunnel lors du transfert d'une session

VPN entre passerelles différentes. Assurer leur synchronisation est un défi qui s'adresse tout au long de cette thèse.



(a) Mono-LAN Architecture (b) Multi-LAN Architecture

Figure B.1: Scenarios de transfert du context IKEv2/IPsec

B.3.2 Management de VPNs du type IPsec

Le transfert d'une connexion VPN d'une passerelle à une autre permet à un client de changer son point d'attachement vers le réseau d'accès. Le principal défi de transférer le trafic VPN concernant une connexion IPsec est de permettre à une SG de transférer tout le matériel cryptographique et les clés dérivées. Nous abordons deux architectures: l'architecture appelée *gestion du trafic* et l'architecture appelée *haute disponibilité (HA)*. Cependant, ces architectures diffèrent dans le fait que la gestion du trafic n'entraîne pas de détection de panne, tandis que l'architecture d'haute disponibilité oui. La gestion du trafic IPsec peut être déclenché lorsque cela est souhaitable, alors que les fonctionnalités de HA sont lancés automatiquement une fois la panne est détectée.

Représentée sur la figure 1.4a, l'architecture mono-LAN illustre le cas où les deux SG ont la même adresse IP. Lors d'un transfert d'un contexte IKEv2/IPsec dans le cas Mono-LAN, il se peut que le client ne soit pas impacté. Le point d'attachement reste pratiquement identique, même s'il s'agit d'une nouvelle SG avec la même adresse IP qui assure la communication. D'autre part, comme représenté sur la figure 1.4b, l'architecture Multi-LAN illustre le cas où deux SGs ont des adresses IP différentes. Cependant, le client est impacté car son point d'attachement a changé d'adresse IP. Comme expliqué ultérieurement, ces modifications peuvent être effectuées en utilisant les extensions de IKEv2: MOBIKE ou REDIRECT. Notez que les scénarios à étudier contemplent le transfer d'un context IKEv2/IPsec, chose qui n'est pas possible avec les standards d'ajourd'hui.

B.4 Pourquoi il y a t-il besoin d'une nouvelle solution?

nous expliquerons d'abord la différence entre le transfer du contexte IKEv2/IPsec et la mobilité IP lors d'une session IPsec active. Un transfert de contexte IKEv2/IPsec consiste à transférer les paramètres négociés (matérielle cryptographique ainsi que les clés dérivées) stockées dans chaque extrémités du tunnel, d'une SG vers une autre. Par contre, la mobilité IP lors d'une session active consiste à mettre à jour l'adresse IP de l'une des extrémités du tunnel VPN, tout en conservant le tunnel entre les mêmes nœuds physiques qu'ont initialement établie la connexion.

Il y a des scénarios où la mobilité IP et le transfert de contexte IPsec/IKEv2 peuvent se produire au même temps. Par exemple, lorsque le transfert de contexte IPsec/IKEv2 a lieu entre deux SG qui possèdent des adresses IP différentes (c'est le cas de l'architecture Multi-LAN), un mobilité IP peut aussi avoir lieu du coté SG. Ceci permettrait de mettre à jour l'adresse IP du tunnel au même temps. Par contre, cette mobilité entraîne que l'utilisateur finale se voit impacté lors du transfert de contexte, car il doit remplacer l'adresse destination du tunnel.

Ils existent trois solutions envisageant la gestion du trafic IKEv2/IPsec. Du coté mobilité IP, un mécanisme appelé MOBIKE [3] permets un noeud de changer son adresse IP destination ou source lors d'une session active d'IPsec. Cependant, ceci concerne la mobilité IP et non le transfert du contexte IKEv2/IPsec. Une déuxieme solution, décrite en [4], introduit une extension d'IKEv2 appelé REDIRECT. Ceci permettrait à une passerelle de rediriger un client vers une autre passerelle. Toutefois, le client est obligé de renégocier tous les paramètres de sécurité à partir de zéro lorsqu'il est redirigé vers une autre passerelle. Cela peut avoir une incidence sur l'expérience du client en raison de retards sur le réseau tout en établissant un nouveau VPN vers une autre passerelle. REDIRECT peut se produire au début de la mise en place d'un tunnel VPN ou au cours d'une session VPN active. D'autre part, REDIRECT ne considère pas le transfert d'un contexte IPsec/IKEv2, ce qui pourrait assurer une meilleure continuité de service et améliorer la qualité de service des utilisateurs.

Une autre solution appelé ClusterIP consiste à créer des clusters basés sur une IP virtuelle commune sans besoin d'avoir un dispositif dédié. Il permet d'implémenter des fonctionnalités de haute disponibilité et d'assurer la connectivité lors de défaillances des passerelles. Le but de ClusterIP est de partager une adresse IP commune afin de construire un cluster de machines pour répartir la charge des paquets IP entrants/sortants. ClusterIP repose sur des règles de pare-feu sous les environnements Linux. Toutefois, afin de mettre en œuvre ClusterIP, il doit être accompagné d'un module de haute disponibilité supplémentaire qui serais capable de synchroniser tous les paramètres IPsec/IKEv2. Le principal inconvénient de ClusterIP est qu'il ne peut pas être déployé entre SG placés dans des différents segments du réseau ou dans des scénarios multi-LANs. C'est une grosse contrainte, parce que les FAI pourraient être intéressés à

	Mono-LAN	**Multi-LAN**
High Availability HA	ClusterIP	Context Transfer + IP Mobility + HA Module
Traffic Management	Context Transfer	Context Transfer + IP Mobility

Table B.1: Solutions for each scenario

avoir plusieurs passerelles avec des adresses IP différentes et répartir géographiquement celles-ci afin qu'ils offrent une meilleure qualité de service aux utilisateurs.

Nous devons donc définir de nouvelles solutions qui permettent la gestion du trafic ainsi que la haute disponibilité pour IPsec/IKEv2. Nous voulons aussi avoir le moins d'impact possible sur les utilisateurs. C'est pourquoi nous devons définir un mécanisme permettant d'effectuer la gestion du trafic entre SGs avec des adresses IP différentes (Multi-LAN).

Le tableau B.1 résume la solution proposé pour chaque scénario. Par exemple, lorsque nous effectuons la gestion du trafic dans une architecture mono-LAN, nous aurons besoin d'effectuer le transfert du contexte afin de maintenir la session VPN active.

B.5 Organisation du Résumé

Le manuscrit est composé de six sections principales dont nous montrons les principales conclusions et perspectives de chaque solution. La section B.6, lequel introduit les notions et les connaissances nécessaires afin de comprendre ce document. Cette description inclue les protocoles IKEv2 et IPsec. Il introduit aussi les notions des différentes mécanismes appelés: REDIRECT, MOBIKE et ClusterIP.

La section **??** identifie les éléments à prendre en compte dans un contexte IKEv2/IPsec. Nous présentons nous premières résultats de nos développements du GET et PUT. Nous illustrons le temps que cela prendre pour récupérer un contexte entier ainsi que le temps de sa réinstallation sur une même passerelle de sécurité.

Les séctions B.8 et B.9 sont dédiées à l'architecture Mono-LAN. D'abord, nous analysons ClusterIP, et nous décrivons comment celui-ci améliore la disponibilité d'un service VPN. En suite nous montrons les résultats de nos tests. D'ailleurs, nous observons les pros et contres de cette solution. Nous allons définir en suite deux architectures pour IKEv2/IPsec: *Haute Disponibilité* et *Gestion du Trafic.*Nous présentons nos implantations à travers de nos fonctions GET et PUT , lesquels sont le cœur du transfert de contexte IKEv2/IPsec.

Dans la section 6, nous nous adressons à des plates-formes VPN dans des environnements multi-LAN. Nous présentons une solution permettant d'effectuer le transfert de contexte entre les différentes passerelles qui possèdent des adresses IP différentes. Nous montrons comment le

client est impacté par la modification de l'adresse IP et la façon de résoudre ce problème ainsi que d'autres facteurs.

B.6 Section 1: L'Etat de l'art

Les FAI utilisent principalement des réseaux privés virtuels (VPN) pour étendre un domaine de sécurité sur des réseaux non sécurisés. Les utilisateurs finaux peuvent sécuriser leurs communications vers leurs fournisseurs d'accès en établissant un tunnel IPsec avec une passerelle de sécurité. Par conséquent, les FAIs sont appelés à offrir des services extrêmement fiables tout en utilisant ces passerelles de sécurité IPsec. D'autre part, un opérateur pourrait vouloir migrer une session IPsec d'une SG à l'autre pour plusieurs raisons (par exemple, prévision de surcharge, la mobilité IP, les défaillance, éviter la ré-authentification, le services de cloud, entre autres).

Compte tenu de l'évolution des réseaux mobiles à travers les dix dernières années, les FAIs doivent faire face à de nouveaux défis en termes de sécurité. Les clions sont de plus en plus soumis à des opérations de mobilité ou Multihoming. En outre, les nouveaux services cloud introduisent de nouveaux défis afin d'offrir une meilleure qualité de service, ainsi que la haute disponibilité de ces contenus, tout en accédant partout où ils le souhaitent.

Tout au long de ce chapitre, nous introduisons les notions de IPsec. Deux entités qui souhaitent établir une communication sécurisée, peuvent utiliser la suite IPsec avec le protocole IKEv2 afin d'authentifier et crypter leurs échanges.

B.7 Section 2: Définition du Contexte

Dans cette partie de la thèse, nous avons présenté le contexte IKEv2/IPsec. Nous avons aussi proposé un mécanisme permettant de extraire et réinstaller dynamiquement un contexte IKEv2/IPsec en utilisant un software appelé strongSwan. Nous avons étudié le temps réel pour établir une session IKEv2/IPsec, ainsi que nous avons mesuré le temps d'extraire et de réinstaller un contexte de sécurité IPsec. Aussi, nous avons mesuré l'impact pour les couches supérieures en générant différentes périodes d'interruption à la couche IPsec lors d'un téléchargement HTTP. Nous montrons l'impact du GET et PUT localement (sur une même passerelle physique). Les sections suivantes seront envisagé de mesurer l'impact du transfert d'un contexte IKEv2/IPsec entre passerelles physiquement différentes. En outre, pour les cas d'utilisation où les différentes SG possèdent des adresses IP différentes, d'autres extensions de IKEv2 (telles que REDIRECT ou MOBIKE) sont étudiées. En plus de l'impacte de bf GET et bf PUT, nous allons examiner les retards ajoutées par les échanges réseau lorsque nous effectuons le transfert du contexte IPsec.

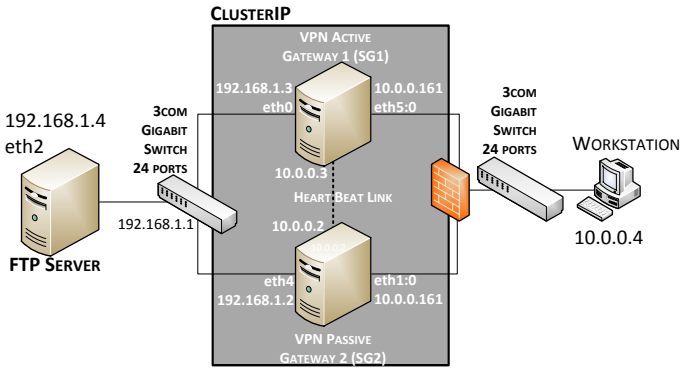
En ce qui concerne un comportement du réseau classique, on considère les résultats des fonctions GET et PUT comme optimistes. Selon les résultats, le pire des cas à effectuer un GET et un PUT sont de 15ms et 22ms respectivement. Cela signifie que nous pouvons rétablir un tunnel dans 37MS (15ms pour obtenir le contexte avec GET + 22ms à effectuer un PUT). Cependant, dans les sections à suivre, nous allons tenir en compte le temps pour transférer le contexte IKEv2/IPsec d'une SG à l'autre. Enfin, pour les scénarios où les SG ont des adresses IP, nous devons prendre en considération le temps de mettre à jour les associations de sécurité du côté du client aussi.

B.8 Section 3: Mono-LAN

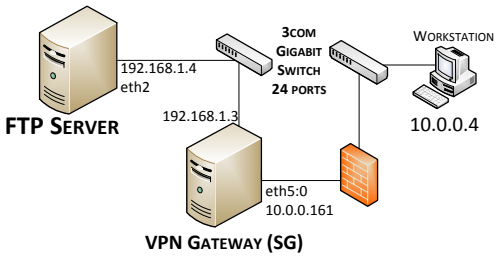
Tout au long de cette section, nous avons mesuré l'impact et la performance à l'aide d'un module de ClusterIP modifié afin de clusteriser différentes SGs avec IPsec. La disponibilité du service IPSec est améliorée grâce à la mise en place d'un system hot-standby assurée par ClusterIP. Aussi, strongSwan garantit la continuité d'une session IPsec grâce à son module HA permettant de diffuser tous les tunnels IPsec entre ses membres du cluster. Les résultats ont montré qu'une SG active dépense de 5% à 8 % de CPU de plus qu'une SG passive lors de l'établissement des tunnels IPsec. Nous avons également constaté qu'il y a une charge supplémentaire lors de l'utilisation du plugin de strongSwan parmi les membres du cluster. Le téléchargement d'un fichier de 500 Mo prenais 20% de plus du temps lors de l'utilisation ClusterIP et le plug-in. C'est pourquoi un administrateur souhaitant mettre en place cette solution doit prendre en compte les coûts de la performance en ajoutant ses fonctionnalités de haute disponibilité à son service VPN. Par ailleurs, un inconvénient principal de ClusterIP est sa limitation à être déployé au sein d'un même segment de réseau. Les sections suivantes se concentreront sur l'utilisation de notre propre outil de gestion de tunnel VPN qui permet le transfert IPsec entre SG dans des environnements mono-LAN. Finalement, on finalisera par le transférer d'un contexte IKEv2/IPsec entre deux SG possédant des adresses IP différentes.

B.9 Section 4: Mono-LAN: L'haute disponibilité et la gestion du trafic VPN

Tout au long de cette section, nous avons décrit le transfert de contexte IKEv2/IPsec dans un environnement mono-LAN, où un cluster offre un service VPN en étant configuré avec une adresse IP commune. Contrairement à la cette section précédente, cette fois nos scénarios comprenaient le transfert de contexte IKEv2/IPsec entre passerelles physiquement différentes. Nous avons défini deux nouvelles composantes: le Heart-Beat et le SYNCH. Ces composants permettent de synchroniser chaque tunnel qui est négocié avec l'un des membres



(a) Scenario using ClusterIP



(b) Scenario NOT using ClusterIP

Figure B.2: Scenarios

du cluster, et le Heart-Beat vérifie régulièrement la activation des nœuds. En introduisant l'architecture haute disponibilité, nous avons ajouté la possibilité de surmonter avec succès un événement basculement grâce à la réplication constante des données IPSec et IKEv2. Nous avons découvert qu'un FAI peut être intéressé par le déploiement de cette architecture afin d'assurer la fiabilité du service VPN. D'autre part, en augmentant la disponibilité du système, l'expérience de l'utilisateur final et la qualité de service est améliorée en évitant les nombreuses ré-authentifications lors du transferts d'une SG à l'autre. En outre, la gestion du contexte IKEv2/IPsec dans un environnement mono-LAN représente des avantages pour le service VPN. En effet, un FAI voulant équilibrer la charge entre les différents membres du cluster peut utiliser cette fonction pour déplacer dynamiquement une session IPsec d'un SG à l'autre. Cela améliore les performances du service VPN et évite les passerelles surchargés au sein du cluster. Nous avons testé deux scénarios différents: la haute disponibilité et la gestion du trafic. Nous considérons également plusieurs paramètres pendant les essais: les méthodes de chiffrement, des transmissions à débit différents ainsi que de protocoles de couches supérieures orientés et non orientés à connexion. Nous avons découvert que le scénario de gestion de contexte est mieux que le scénario de HA. En fait, même si les deux scénarios prévoient le transfert d'un contexte IKEv2/IPsec, ils se comportent différemment et pourraient être mises en place dans des circonstances différentes. La composante Heart-Beat du scénario HA ajoute automatisme pendant les failles, mais le transfert du contexte IKEv2/IPsec n'arrive qu'une fois le Heart-beat donne l'ordre de le faire.

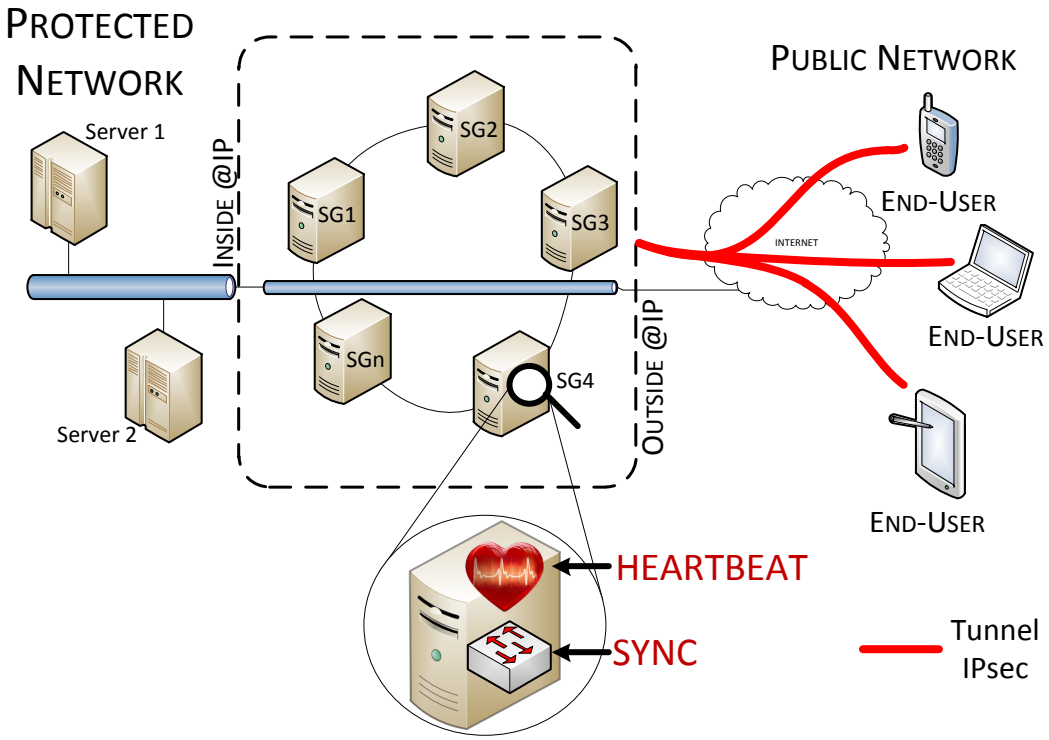


Figure B.3: High Availability scenario for n gateways

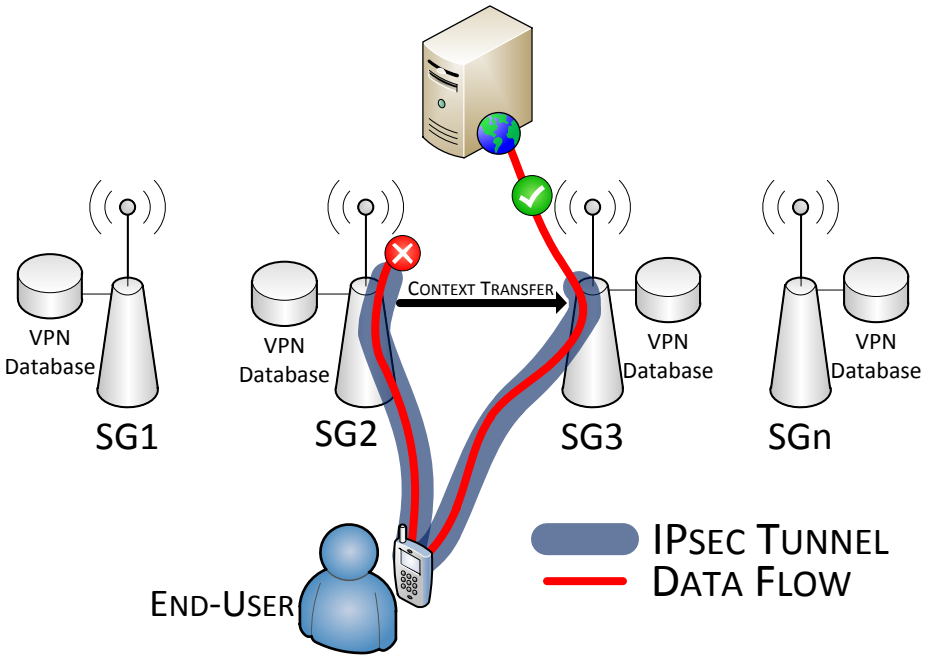


Figure B.4: Context Management scenario

High Availability	Reference QoS ($^0/_{8s}$)	$^1/_{8s}$	$^2/_{8s}$	$^3/_{8s}$	$^4/_{8s}$
HTTP-8Kbps-CBC	100%	+3%	+6%	+17%	+17%
HTTP-8Kbps-CTR	100%	0%	+19%	+19%	+19%
HTTP-48Kbps-CBC	100%	-50.5%	-68%	-70%	-69%
HTTP-48Kbps-CTR	100%	-52%	-62%	-69%	-69%
HTTP-96Kbps-CBC	100%	-27%	-67%	-71%	-69%
HTTP-96Kbps-CTR	100%	-60%	-63%	-71%	-72%
RTSP-8Kbps-CBC	100%	+14%	+31%	+38%	+34%
RTSP-8Kbps-CTR	100%	+9%	+39%	+39%	+30%
RTSP-48Kbps-CBC	100%	-40%	-52%	-49%	-51%
RTSP-48Kbps-CTR	100%	-60%	-63%	-62%	-61%
RTSP-96Kbps-CBC	100%	-57%	-60%	-59%	-60%
RTSP-96Kbps-CTR	100%	-56%	-58%	-57%	-58%

Table B.2: Impacte de la fréquence des évènements de HA sur la QoS

Au lieu de cela, dans un scénario de gestion du trafic, un FAI est capable de déplacer une session IPsec en cas de besoin. Cela améliore la performance et l'expérience de l'utilisateur final. C'est la principale raison pour laquelle le scénario de gestion de trafic a des résultats plus satisfaisants.

Nos résultats sont présentés en termes de comportement du réseau et du QoS. Pendant les essais de HA, nous avons remarqué que le comportement du réseau des protocoles orientés non-transmission (par exemple UDP) sont moins impactés par un événement de basculement qu'un protocole orienté à connexion (tel que TCP). En fait, en raison de la nature de TCP qui impose de accuser les paquets transmis, ceci génère des retards supplémentaires pour récupérer les basculement. Cependant, d'un point de vue QoS, TCP a obtenu de meilleurs résultats que UDP. En effet, avec UDP, les paquets qui sont perdus lors des événements de basculement ne sont pas récupérés, dégradant ainsi la qualité de service. Concernant les résultats de la gestion du trafic, nous avons remarqué que les protocoles orientés connexion sont moins impactés par rapport aux scénarios où les fonctionnalités HA sont actives. Ceci est principalement dû au temps que prend le Heart-Beat d'accomplir sa tâche. En fait, au cours d'une gestion de contexte, il n'y a plus de Heart-Beat impliqué lors du transfert du contexte IKEv2/IPsec, et donc, le SG qui prends la responsabilité d'une connexion n'a pas besoin d'attendre de vérifier la vivacité de la passerelle et donc d'installer la nouvelle session IPsec.

Les futurs travaux comprennent l'interaction de IPsec sur un transfert de contexte entre SG avec des adresses IP différentes. Même si nous avons amélioré le service VPN par la création du cluster de nœuds avec une adresse IP unique, un FAI peut également être intéressé par le déplacement d'une session IPsec entre SG distants avec des adresses IP différentes. Nous pensons que cela peut être avantageux d'assurer la continuité d'un service VPN pour un utilisateur mobile qui est probablement en train de changer son point d'attachement. Ceci est expliqué en détail dans section suivante.

Context Management	Reference QoS ($^0/_{8s}$)	$^1/_{8s}$	$^2/_{8s}$	$^3/_{8s}$	$^4/_{8s}$
HTTP-8Kbps-CBC	100%	0%	+1%	0%	+1%
HTTP-8Kbps-CTR	100%	0%	0%	+5%	+1%
HTTP-48Kbps-CBC	100%	-18%	-20%	-40%	-28%
HTTP-48Kbps-CTR	100%	-41%	-31%	-39%	-33%
HTTP-96Kbps-CBC	100%	-28%	-16%	-40%	-48%
HTTP-96Kbps-CTR	100%	-33%	-20%	-53%	-50%
RTSP-8Kbps-CBC	100%	0%	+16%	+21%	+28%
RTSP-8Kbps-CTR	100%	+3%	+15%	+23%	+28%
RTSP-48Kbps-CBC	100%	-31%	-41%	-50%	-52%
RTSP-48Kbps-CTR	100%	-35%	-57%	-58%	-61%
RTSP-96Kbps-CBC	100%	-53%	-61%	-66%	-68%
RTSP-96Kbps-CTR	100%	-27%	-43%	-49%	-49%

Table B.3: Impacte de la frequence des évènements de Context Management sur le QoS

B.10 Section 5: Multi-LAN

Tout au long de cette section, nous présentons la façon de procéder lors de l'exécution d'un transfert de contexte d'une session IKEv2/IPsec basé sur une extension de la mobilité, MOBIKE. Cette solution peut avoir des impacts positifs concernant l'expérience d'un client à chaque fois qu'une session VPN est transférée entre différents dispositifs. En fait, comme nous l'avons vu dans les sections précédents, une session IPsec est initialement destiné à être maintenu dans le même dispositif qui a établi la connexion. Cependant, d'un point de vue FAI, il peut y avoir des avantages à transférer une session IPsec entre différents SGs.

Nous avons donc défini le scénario Multi-LAN comme dans la figure B.6. Nous avons également décrit les échanges lors du transfert de contexte IKEv2/IPsec dans la figure B.5. Nous nous attendons à réduire la quantité de calcul et le temps que les FAIs dépense pendant les établissements d'un tunnel. En effet, la consommation d'énergie des FAIs serait réduit tant que les IKE_SAs et IPsec_SAs soient conservés lors du transfert d'un contexte IKEv2/IPsec. Cela améliore le comportement réel du réseau en termes d'énergie et de gestion du trafic.

Nous avons détecté la différence entre le mode de transport et le mode tunnel IPsec. En mode tunnel, le transfert de contexte est pratiquement transparent pour les clients. En effet, le trafic IPsec (ESP), est effectivement tunnelé dans une nouvelle payload contenant un nouvel en-tête IP externe (l'en-tête IP interne reste le même). Toutefois, lorsque IPsec est utilisé dans le mode de transport, les implémentations des couches supérieures doivent faire face à la mobilité IP. Ainsi, comme l'en-tête IP du trafic IPsec change aussi, nous avons effectué quelques recherches en termes de mobilité pour le mode de transport IPsec, où nous avons conçu un protocole appelé MOBIKE-X.

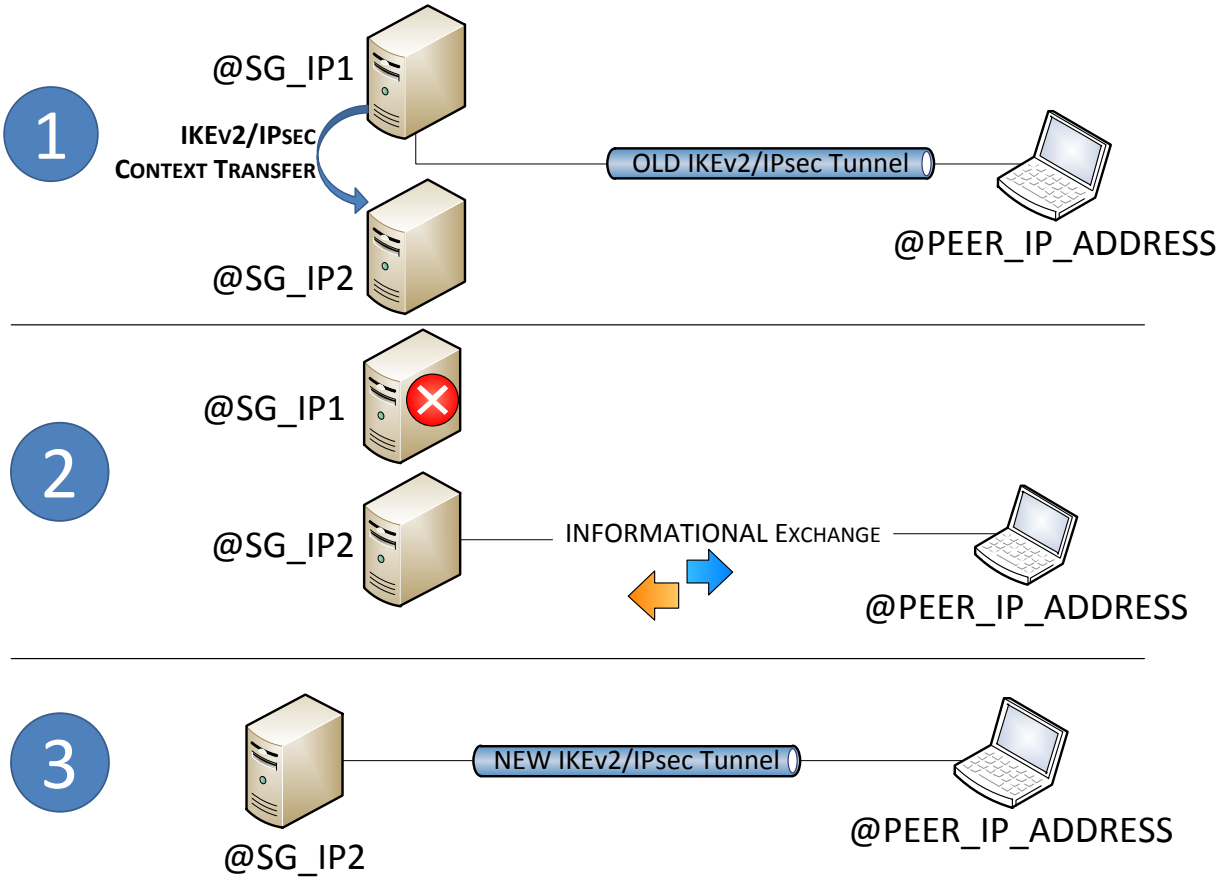


Figure B.5: Multi-LAN IKEv2/IPsec context transfer

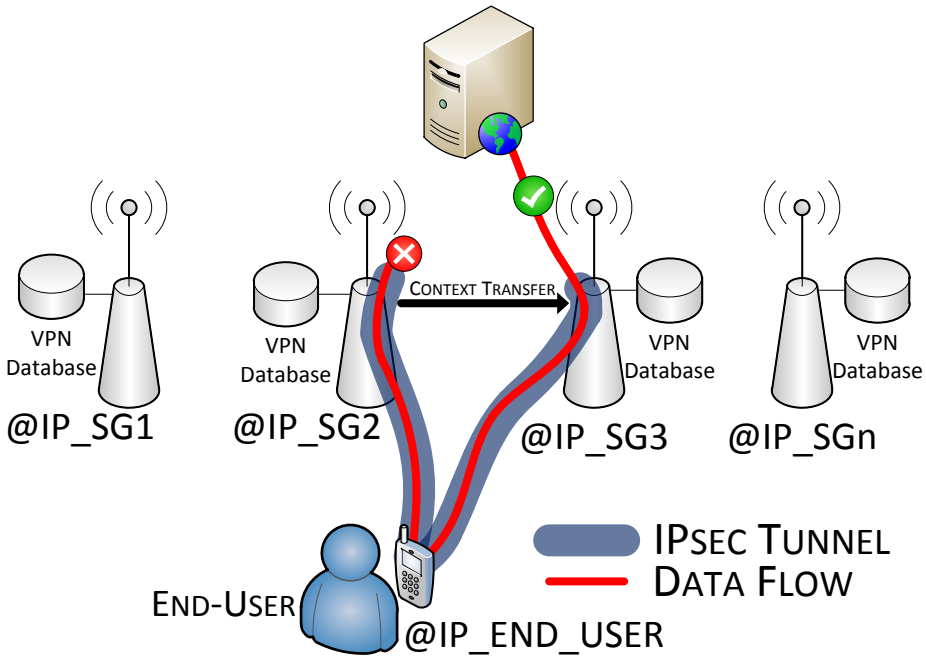


Figure B.6: Scenario of a Multi-LAN IKEv2/IPsec context transfer

Les futurs travaux comprennent la mise en place du scénario Multi-LAN. Une nouvelle prolongation du contrat de recherche a été approuvé mutuellement entre Orange (France Telecom) et l'Université de Paris 6, afin de poursuivre ce thème de recherche.

B.11 Conclusions et Perspectives

Cette thèse se concentre à garantir et améliorer la continuité de service de communications basés sur IKEv2/IPsec, principalement mis en œuvre dans les services VPN.

La première partie de notre investigation est porté sur la façon de protéger les réseaux IP avec IPsec. Nous avons également introduit le concept de l'utilisation IKEv2 comme le complément d'IPsec afin de procéder à l'authentification des nœuds et de la gestion ultérieure des connexions IPsec. Cela permet aux opérateurs d'offrir un environnement fiable et sécurisé sur un réseau non sécurisé. En outre, en fournissant une extension IKEv2 du type MOBIKE ou REDIRECT, nous prétendons présenter un cadre pour le transfert de contexte transparent afin de gérer dynamiquement une session VPN.

Les études ultérieures nous sont amené à développer deux nouvelles fonctionnalités concernant IKEv2/IPsec VPN: les fonctions GET et PUT. Notre motivation principale était de parvenir à une gestion dynamique d'une session VPN. Nous avons trouvé que l'extraction de toutes les informations concernant une session IKEv2/IPsec prendrait environ 15ms, et que réinstaller ceux-ci prendrait environ 22ms. Nos développements ont été effectué sous l'implémentation Open Source de *StrongSwan*.

La deuxième partie de la thèse a été consacrée à l'environnement Mono-LAN. Nous avons étudié la mise en œuvre de ClusterIP, un mécanisme permettant de créer un cluster de SGs configurés sous une même adresse IP virtuelle. Les résultats dans cette partie on montré qu'une SG actif dépense 5 % à 8 % de CPU plus qu'une SG passive lors du regroupement des tunnels IPsec avec le mécanisme ClusterIP. Nous avons également constaté qu'il y a une charge supplémentaire lors de l'utilisation du HA-plugin de strongSwan parmi les membres du cluster. D'autres investigation nous sont amenés à fusionner ClusterIP avec nos propres fonctions (GET et PUT). Nous avons créé un cadre de transfert de contexte IKEv2/IPsec transparente pour les clients. Cela nous a conduit à la définition de deux nouvelles architectures offrant des capacités différentes: gestion du trafic IPsec et l'haute disponibilité. Ceux-ci sont basés sur nos implémentations GET et PUT. Nous avons présenté nos résultats en termes de comportement du réseau et de la qualité de service. Lors des expérimentes sous l'architecture de HA, nous avons constaté que le comportement du réseau des protocoles non-orientés à connexion (par exemple, UDP) sont moins impactés par le basculement qu'un protocole orienté à connexion (par exemple, TCP). En ce qui concerne les résultats du gestion du trafic, nous avons remarqué qu'en générale, cette architecture est moins impactée que les scénarios où les fonctionnalités

HA sont activés. Tout au long de la deuxième partie de la thèse, nous remarquons que ce type de mécanismes permettrait d'effectuer de l'équilibrage de charge des sessions VPNs, ainsi que d'effectuer de la répartition de charge entre les différents membres du cluster (sans besoin d'avoir un dispositif dédié pour cela). Cependant, nous avons également constaté que ces solutions sont plus spécifiques pour des groupes de nœuds situé localement, où les SG sont placés physiquement proche les unes des autres, restant configurés dans un même segment de réseau.

En suite, la troisième partie de la thèse a été dédiée aux environnements multi-LAN. Nous avons défini une architecture pour le transfert des contextes IKEv2/IPsec entre SGs possédant des adresses IP différentes. Bien que les deux modes d'IPsec, le mode tunnel et le mode transport, impliquent une certaine signalisation IKEv2 (par un échange INFORMATIONAL) lors d'un changement du point d'attachement, ils sont impactés différemment. Dans le mode tunnel, lorsque l'adresse IP associée au l'une des extrémités du tunnel change, cette interruption est complètement transparente pour les applications des couches supérieures. Même si l'adresse IP externe d'un tunnel change, l'adresse IP interne du tunnel reste le même. Par contre, pour les communications en mode transport, il n'y a pas d'adresse IP interne. Les couches supérieures devront faire face à la mobilité IP à chaque fois que son adresse IP associé à une IKE_SA et IPSec_SA change. Nos efforts se concentrent à permettre l'opérateur à déplacer un client à partir d'une SG vers une autre. Tout cela, en utilisant une signalisation minimale, moins de consommation et moins de calcul associée à la construction du matériel cryptographique.

Evolution Globale et travail à venir

Comme la plupart des terminaux sont à nous jours équipés avec des interfaces réseau d'accès alternatifs (par exemple WiFi), nous pensons que les opérateurs doivent se concentrer sur la façon dont ils peuvent être utilisés pour accéder aux services. D'autre part, l'une des évolutions possibles pour permettre la circulation à partir de réseaux d'accès mobiles est de décharger le trafic réseau d'accès radio (RAN) vers des réseaux d'accès complémentaires tels que le WIFI, WIMAX, femtocells, etc. Ces réseaux sont appelés à maintenir le même niveau de sécurité aussi bien que les RANs. En outre, les réseaux d'accès non mobiles, comme le WiFi, sont très répandus comme des extensions de réseaux sans fil à haut débit, principalement dans les espaces intérieurs où les technologies cellulaires ne fonctionnent pas bien.

Les FAI sont intéressés à offrir de nouvelles fonctionnalités à ses services VPNs. Même si le contenu multimédia d'aujourd'hui sont principalement offerts par les services Web (HTTP streaming, VoIP, etc), l'accès au réseau lui-même est toujours géré par les FAI. Les opérateurs doivent être capables de s'adapter facilement à de nouvelles opportunités. Par exemple, en sachant ses choix au niveau de l'architecture du réseaux, un FAI tel qu'Orange peut proposer NaaS (réseau en tant que service) des services à l'avenir, ce qui nécessite la mise en œuvre des VPN. Par conséquent, l'amélioration de la fiabilité, la disponibilité et la qualité de ces services est une question d'intérêt.

Comme indiqué tout au long de cette thèse, le trafic de données mobiles est en croissance permanente et annuelle (60% à 70%). Mis à part de la croissance du trafic, il y a eu aussi la généralisation des plans de facturation forfaitaire, une plus de la grande disponibilité de contenus attractifs ainsi que la la disponibilité croissante des appareils de grande consommation (par exemple, ordinateurs portables, smartphones, tablettes, etc.). Tout ce contexte crée une forte pression sur les opérateurs pour qu'ils donnent la meilleur performance possible de leurs réseaux.

Nos propres développements dans ce domaine nous permettent de gérer un tunnel VPN. Ceux-ci permettent à un opérateur de gérer dynamiquement le EUs qui utilisent ses services VPN et avoir un impact positif sur leurs expérience et qualité de service. Cependant, il reste encore du travail afin de faire integrer ceux-ci à l'intérieur des réseaux opérationnelles.

Tout d'abord, même si nos bancs d'essai concernant les environnements mono-LAN considérais le transfert du contexte IKEv2/IPsec entre deux SG différentes, nous croyons qu'il est important de définir un algorithme qui décide la responsabilité des paquets entrants vers un cluster composé par n nœuds. Pour les scénarios de haute disponibilité, nous avons abordé les scénarios actifs/passifs, mais pour le partage de charge ou des scénarios d'équilibrage de charge, il serait important de définir cet algorithme qui déciderait comment répartir la charge entre les différents membres du cluster.

En ce qui concerne les environnements multi-LAN, nous avons présenté une solution basée sur MOBIKE permettant d'effectuer le transfert de contexte. Cette solution résout le cadre d'un contexte IKEv2/IPsec entre les nœuds avec une adresse IP différente. Comme la période de la thèse touchais à sa fin, cette mise en place fait partie des travaux futurs de ma thèse. Cependant, un projet très motivant émerge afin de poursuivre nos recherches dans ce domaine. Une fois les développements de Multi-LAN sont faites, nous prétendons donner à la communauté un mécanisme qui améliore le comportement du réseau. En fait, les futurs réseaux ont tendance à être plus hétérogène, et ce mécanisme permettrait de transférer une session VPN entre différents dispositifs. En outre, on prétend réduire les retards de rétablir un nouveau tunnel VPN vers un nouveau dispositif, au lieu de transférer ces données entre les dispositifs entre eux-même.

Il peut être important de concentrer les efforts à définir une sorte d'intelligence locale afin de gérer les sessions IKEv2/IPsec. Une sorte d'application capable d'externaliser la décision de la gestion des sessions. Par exemple, un cas d'utilisation serait un gestionnaire de connexions afin de décider d'effectuer un transfert de contexte, des fonctions de mobilité ou multihoming, ou de gérer dynamiquement les aspects de sécurité. Comme IKEv2 est le protocole permettant de négocier des modifications à IPsec, le gestionnaire de connexions serait en mesure de donner des ordres afin de personnaliser une session IKEv2/IPsec active.

Aujourd'hui, les réseaux sont composés de plusieurs technologies d'accès, où chacun définit

ses propres normes de sécurité, introduisant plus d'hétérogénéité dans le réseau Internet. Cette situation présente l'inconvénient qu'il est complexe à maintenir le même niveau de sécurité entre les différentes technologies d'accès au réseau, ce qui peut inclure différents appareils (constructeurs). Cependant, les communcation de radio ont désormais migré vers les communications en IP. Cela nous permet de proposer IPsec en tant que mécanisme de sécurité sur tout le réseau. Notre intérêt est de poursuivre nos investigations dans le but d'introduire des mécanismes plus sophistiqués qui permettent de maintenir le même niveau de sécurité indépendamment de la technologie d'accès. Nous prétendons proposer un projet au sein de l'IETF afin de pousser la standardisation vers le transfert de contexte IPsec.

Accomplished Work

Publications

The results of this thesis, are reflected in some publications and contributions. We are also currently working on publishing the code for the open source community. Publications include:

- D. Palomares, D. Migault, and M. Laurent, "Failure Preventive Mechanism for IPsec Gateways", in *International Conference on Communications and Information Technology* - ICCIT 2013, Beirut - Lebanon.

- D. Palomares, D. Migault and M. Laurent "Mechanisms to ensure continuity of service for IKEv2/IPsec based communications", in *International Conference on Secure Networking and Applications* - ICSNA 2011, Paris - France.

- D. Palomares, D. Migault, W. Velasquez and M. Laurent "High Availability for IPsec VPN Platforms - ClusterIP Evaluation", in *8th International Conference on Availability, Reliability and Security* ARES 2013, Regensburg - Germany.

Contributions as co-author

- D. Migault, D. Palomares, E. Herbert, W. You, G. Ganne, G. Arfaoui and M. Laurent, "E2E: An Optimized IPsec Architecture for Secure And Fast Offload", in *7th International Conference on Availability, Reliability and Security*, ARES 2012, Prague - Czech Republic.

- Migault, D., Palomares, D., Herbert, E., You, W., Arfaoui and Laurent, M, "ISP Offload Infrastructure to minimize cost and time deployment", GLOBCOM 2012, Anaheim, CA, USA.

- [32, 37, 38] are contribution for MOBIKE-X at the IETF.

Pieces of Software

Source code includes:

- GET and PUT functions. This implementation is based on strongSwan 4.5.0.
- IPsec-CTX form guide lines for Mono-LAN environments on a Wiki.
- A set of scripts used to perform context transfer tests for all the testbeds presented throughout this thesis.
- A webpage with some tools and explanation of our work (this web site is currently under construction).

Acronyms and Definitions

EU End User

HA High Availability

IKEv2 Internet Key Exchange, version 2

IKE SA IKE Security Association

IPsec Internet Protocol security

IPsec SA IPsec Security Association

IPsec-CXT IPsec context transfer

MN Mobile Node

MIP Mobile IP

SA Security Association

SG Security Gateway

VPN Virtual Private Network

WLAN Wireless Local Area Network

Bibliography

[1] Cisco. (2013) Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005, updated by RFC 6040. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[3] P. Eronen, "IKEv2 Mobility and Multihoming Protocol (MOBIKE)," RFC 4555 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4555.txt

[4] V. Devarapalli and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 5685 (Proposed Standard), Internet Engineering Task Force, Nov. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5685.txt

[5] M. Georgiades, H. Wang, and R. Rafazolli, "Security of context transfer in future wireless communications," in *Wireless World Research Forum (WWRF)*, Toronto, Canada, Nov 2004.

[6] S. Ayaz, C. Bauer, and M. Ehammer, "Applying IKE/IPsec Context Transfer to Aeronautical Networks," in *7th ACM international symposium on Mobility management and wireless access (MobiWAC '09)*, 2009.

[7] J. Loughney, M. Nakhjiri, C. Perkins, and R. Koodli, "Context Transfer Protocol (CXTP)," RFC 4067 (Experimental), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4067.txt

[8] F. Allard, "Le transfert de contexte : atout pour la mobilité et outil de réduction des coûts pour la sécurité," Ph.D. dissertation, RSM - Dépt. Réseaux, Sécurité et Multimédia (Institut Mines-Télécom-Télécom Bretagne-UEB), 2009.

[9] V. Devarapalli and F. Dupont, "Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture," RFC 4877 (Proposed Standard), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4877.txt

[10] F. Allard and J.-M. Bonnin, "An application of the context transfer protocol : IPsec in a IPv6 mobility environment," *International journal of communication networks and distributed systems*, vol. 1, no. 1, pp. 110 – 126, 2008.

[11] F. Allard, J.-M. Bonnin, J.-M. Combes, and J. Bournelle, "IKE Context Transfer in an IPv6 Mobility Environment," in *MobiArch'08, 22 août, Seattle (WA), USA*. FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), RSM - Dépt. Réseaux, Sécurité et Multimédia (Institut Mines-Télécom-Télécom Bretagne-UEB), 2008.

[12] F. Allard, J.-M. Combes, R. M. Lopez, and A. G. Skarmeta, "Security Analysis and Security Optimizations for the Context Transfer Protocol," in *NTMS'08 : second International Conference on New Technologies, Mobility and Security, 5-7 novembre, Tanger, Maroc.* FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), UMU - University of Murcia, 2008.

[13] F. Allard and J.-M. Bonnin, "An application of the context transfer protocol : IPsec in a IPv6 mobility environment," in *BWIA'07 : International workshop on Broadband wireless Internet access, 21 août, Ottawa (ON), Canada.* FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), RSM - Dépt. Réseaux, Sécurité et Multimédia (Institut Mines-Télécom-Télécom Bretagne-UEB), 2007.

[14] F. Allard, "Etude de faisabilité concernant le transfert de contexte pour la sécurité," in *GRES'06, 9-12 mai, Bordeaux, France.* FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), 2006.

[15] R. Singh, G. Kalyani, Y. Nir, Y. Sheffer, and D. Zhang, "Protocol Support for High Availability of IKEv2/IPsec," RFC 6311 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6311.txt

[16] A. Steffen., "the OpenSource IPsec-based VPN Solution: *StrongSwan.*" [Online]. Available: http://www.strongswan.org

[17] POLQA, "Perceptual Objective Listening Quality Assessment - POLQA." [Online]. Available: http://www.polqa.de

[18] D. Palomares, "Mechanisms to Ensure Continuity of Service for *IPsec/IKEv2* Based Communications," in *ICSNA-2011: International Conference on Secure Networking and Applications ,24-25 October, Paris, France.* FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), 2011.

[19] D. Palomares, D. Migault, and M. Laurent, "Failure Preventive Mechanism for IPsec Gateways," in *International Conference on Communications and Information Technology - ICCIT'13*, 2013.

[20] D. Migault, D. Palomares, E. Herbert, W. You, G. Ganne, G. Arfaoui, and M. Laurent, "E2E: An Optimized IPsec Architecture for Secure And Fast Offload," in *International Workshop on Security of Mobile Applications IWSMA'12 (co-located with ARES'12)*, Aug. 2012.

[21] D. Migault, D. Palomares, E. Herbert, W. You, G. G. G. Arfaoui, and M. Laurent, "ISP Offload Infrastructure to minimize cost and time deployment," in *Proceedings of IEEE Global Telecommunications Conference - Communication and Information System Security (GLOBECOM '12)*, Dec. 2012.

[22] ITU, *Security architecture for Open Systems Interconnection for CCITT applications (ITU-T Recommendation X.800)*, International Telecommunications Union, Mar. 1991.

[23] S. Kent, "IP Authentication Header," RFC 4302 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4302.txt

[24] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, obsoleted by RFCs 4303, 4305. [Online]. Available: http://www.ietf.org/rfc/rfc2406.txt

[25] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306 (Proposed Standard), Internet Engineering Task Force, Dec. 2005, obsoleted by RFC 5996, updated by RFC 5282. [Online]. Available: http://www.ietf.org/rfc/rfc4306.txt

[26] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 5996 (Proposed Standard), Internet Engineering Task Force, Sep. 2010, updated by RFC 5998. [Online]. Available: http://www.ietf.org/rfc/rfc5996.txt

[27] D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, obsoleted by RFC 4306. [Online]. Available: http://www.ietf.org/rfc/rfc2407.txt

[28] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, obsoleted by RFC 4306. [Online]. Available: http://www.ietf.org/rfc/rfc2408.txt

[29] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, obsoleted by RFC 4306, updated by RFC 4109. [Online]. Available: http://www.ietf.org/rfc/rfc2409.txt

[30] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba, and F. Bersani, "The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method," RFC 5106 (Experimental), Internet Engineering Task Force, Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5106.txt

[31] M. Devera, "HTB Linux queuing discipline manual - user guide." [Online]. Available: http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

[32] D. Migault, "MOBIKE eXtension (MOBIKE-X) for Transport Mobility and Multihomed IKE_SA," (Work in Progress), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/html/draft-mglt-ipsec-mm-mobikex-00

[33] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC 6071 (Informational), Internet Engineering Task Force, Feb. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6071.txt

[34] 3GPP-LTE, "3gpp system to wireless local area network (wlan) interworking; system description, ts 23.234, release 10," Standard, Mar. 2011.

[35] L. Yu, S. Jia, C. Xu, J. Guan, and D. Gao, "An ipsec seamless switching mechanism with high availability and scalability by extending ikev2 protocol," *IET Conference Publications*, vol. 2011, no. CP588, 2011.

[36] Y. Nir, "IPsec Cluster Problem Statement," RFC 6027 (Informational), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6027.txt

[37] D. Migault, "IPsec mobility and multihoming requirements : Problem statement," (Work in Progress), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/html/draft-mglt-ipsec-mm-requirements-00

[38] D. Migault and C. Williams, "Multiple Interfaces IPsec Security Requirements," (Work in Progress), Internet Engineering Task Force, Mar. 2012. [Online]. Available: https://datatracker.ietf.org/doc/draft-mglt-mif-security-requirements/

[39] Wikipedia, "Boxplot — Wikipedia The Free Encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Box_plot

[40] J. Kempf, "Problem Description: Reasons For Performing Context Transfers Between Nodes in an IP Access Network," RFC 3374 (Informational), Internet Engineering Task Force, Sep. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3374.txt

[41] R. Hinden, "Virtual Router Redundancy Protocol (VRRP)," RFC 3768 (Draft Standard), Internet Engineering Task Force, Apr. 2004, obsoleted by RFC 5798. [Online]. Available: http://www.ietf.org/rfc/rfc3768.txt

[42] R. Housley, "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)," RFC 3686 (Proposed Standard), Internet Engineering Task Force, Jan. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3686.txt

[43] D. W. Helger Lipmaa, Phillip Rogaway, "Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption," in *Workshop*. University of California Berkeley, 2010.

[44] VLC, "VLC streaming protocols." [Online]. Available: http://www.videolan.org/streaming-features.html

[45] Arecord, "Arecord native ALSA command." [Online]. Available: http://alsa.opensrc.org/Arecord

[46] J. P. Richard Reynolds, "Application Guide for Recommendation ITU T P.863." [Online]. Available: http://www.itu.int/ITU-T/workprog/wp_item.aspx?isn=8909