Dynamic setup of IPsec VPNs in Service Function Chaining

[1]Håkon Gunleifsen[a,*], Thomas Kemmerich[1,a], Vasileios Gkioulos[1,a]

[a]*Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU), Postbox 191, 2802 Gjøvik, Norway*

Abstract

This article describes a novel mechanism for the automated establishment of dynamic Virtual Private Networks (VPN) in the application domain of Network Function Virtualization (NFV). Each hop in an NFV Service Function Chain (SFC) lacks the capability of per-flow encryption, that makes the traffic flow in federated NFV environments vulnerable for eavesdropping. Due to the possible lack of bidirectional data plane communication channels between VNFs in an SFC, the Internet Security Key Exchange protocol (IPsec-IKE) is not applicable inside a VNF. Hence, this article introduces an alternative to IPsec-IKE that is specifically designed for NFV environments. This component is named Software Defined Security Associations (SD-SA), which is shown through a proof of concept evaluation to perform better than IPsec-IKE with respect to bandwidth and resource consumption.

Keywords: NFV; SFC; NSH; IPsec; IKE; SD-IKE; RESTconf

1. Introduction

The security mechanisms in Software Defined Networks (SDN) and NFV lack the capability to encrypt and isolate the end-user traffic between VNFs. Figure 1 exemplifies the problem by describing a typical VNF Service Function Chain (SFC), where the network traffic traverses multiple VNFs located at multiple service providers, while earlier work [1, 2, 3] shows that the current NFV standardization attempts from ETSI [4] and IETF [5] do not take VNF isolation into account in the SFC design.

Accordingly, an end-user who subscribes to VNF services from multiple services providers:

- Cannot end-to-end encrypt traffic, since the VNFs require to have access in order to manipulate this traffic.
- Is not aware and in control of which service providers having access to the data traffic, can potentially eavesdrop traffic and manipulate the route tables.
- Does not know if the VNFs are shared network services with other users, who can as such access private data.

*I am the corresponding author

Email addresses: hakon.gunleifsen2@ntnu.no ([1]Håkon Gunleifsen), thomas.kemmerich@ntnu.no (Thomas Kemmerich), vasileios.gkioulos@ntnu.no (Vasileios Gkioulos)

Earlier work [2], [1], showed that these problems could be resolved by introducing hop-by-hop encryption per IP flow or per group of IP flows. This is enabled by deploying an encryption application in front of every VNF within an SFC (Figure: 1). We showed that this encryption application is typically a Virtual Machine attached to the Virtual Link [6], particularly assigned for this function. These underlying encryption functions [1] can also be perceived as regular VNFs. Furthermore, earlier work also showed an additional problem with such an architecture. A service chain, following the NSH and SFC specification [6], can have a different service path than the reverse service path. Consequently, a pair of encrypting and decrypting VNFs in a service chain, do not necessarily have a bidirectional communication channel on the data plane, where they can exchange keys. Hence, there is a need for a new key exchange mechanism that is not dependent on a point-to-point bidirectional communication channel. This particular lack of a data plane communication channel and the need for flow-based encryption is specific to the application domain of NFV. In this article we continue this work, focusing on the authentication and key distribution, seeking to automate the set up of secure channels between VNFs.

The investigated research problem is similar in nature to the auto-configuration of VPN setups [7], while based on the reviews of RFC 7018 [7], the earlier lack of use cases for such a protocol might be the reason for this not being resolved. Yet, the emergence of SDN and NFV technologies, highlight security use cases that necessitate renewed effort towards this direction. Accordingly, a similar problem was also stated in a recent Internet draft regarding the VNF registration process over the Interface to Network Security Functions (I2NSF) [8],[9]. The draft shows that automation of Network Security Functions such as VPNs is challenging. This is due to the lack of a secure key distribution mechanism and the lack of support for multi-vendor and multi-operator use cases (I2RS [10]). This problem is resolved by the solution presented in this article, by having a separate SDN controller handling all key distributions in a multi-operator SFC.

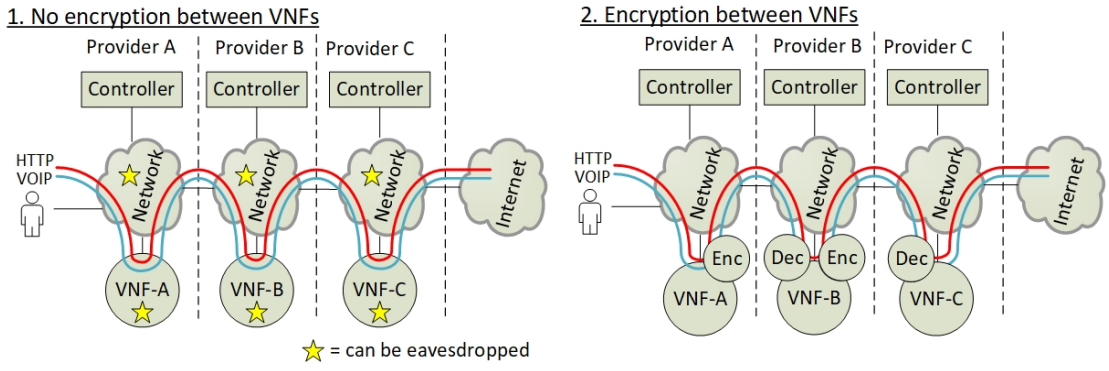

Figure 1: Use case and possible adversarial placement

Figure 2 shows how the network topology can be simplified for the use case described in Figure: 1, where it is assumed that one orchestration plane is capable of orchestrating the distribution of tunnel connection parameters and keys. The simplified figure shows how

an IP packet from the end-user is routed through a network, with Network Service Header (NSH) [11] transport and encryption enabled per flow. Accordingly, an encrypted tunnel per flow between every VNF can ensure that end-user traffic traversing an SFC can only be accessed by the related VNFs, assuring that only these VNFs have the encryption keys to access this component of the data-flow.

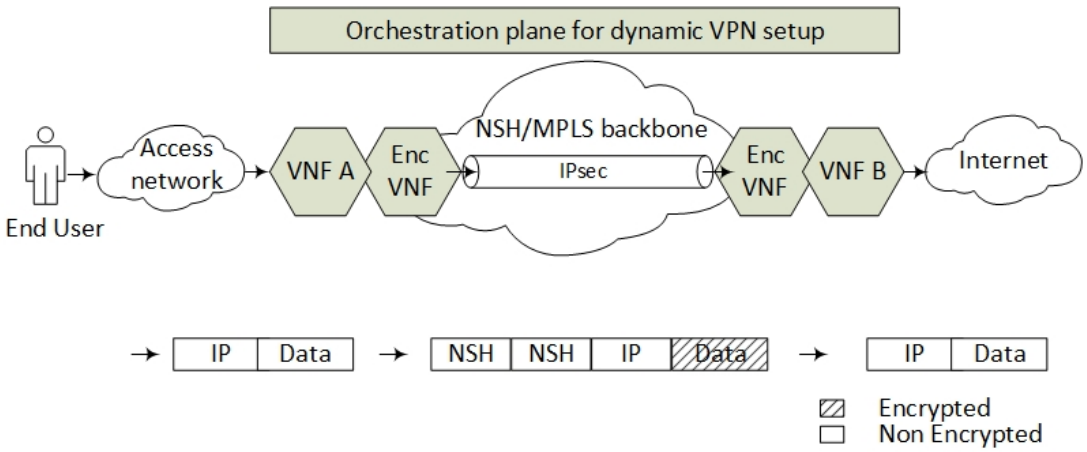

Figure 2: Network topology simplification

This paper introduces a mechanism for the isolation and encryption of data traffic between VNFs in a federated NFV environment. We introduce a method for mutual and secure authentication of encryption functions in an SFC in order to establish a secure channel between them. An architecture of a site-to-site VPN setup with a new mechanism to distribute both initial keys and cipher keys is presented, while in addition to the theoretical aspect of the new protocol, the paper also presents the empirical results from experiments on the implemented design.

The main contributions in this article are:

- A set of requirements for NFV services running isolated services.
- A new architecture of a key exchange mechanism in distributed NFV environments.
- A performance and security analysis of the security mechanism proposed.

The remainder of this article is structured as follows. The most related work concerning VPN authentication follows this introduction. Section 2 defines the prerequisites, constraints, topology assumptions and requirements that are needed in order to apply the new authentication mechanism. The section also correlates the requirements with existing alternatives. The architecture in Section 3 suggests a design for automating VPN configurations, while Section 4 shows how this is implemented. Section 5 demonstrates a proof of concept experiment with performance tests. An evaluation of the performance test results is presented in Section 6, while Section 7 concludes this paper.

1.1. Related work

NFV allows Internet Service Providers to provide flexible network service deployments. However, recent research [12] have shown that this new technology should be secured from

both the service provider and the end-user perspective. NVF surveys [13], [14], [15] have pointed out multiple threats and vulnerabilities in NFV, where end-user privacy is an open issue. Due to the fact that VNFs are acting as middleboxes and require access to the data-content, makes end-to-end encryption difficult. Multi-Context TLS [16] was developed to solve this issue. However, the protocol is insecure [17] and it provides neither flow-based encryption nor SFC isolation. Hence, another approach to this problem is to enable the NFV infrastructure to provide hop by hop encryption and automatically exchange keys and set up secure channels between the VNFs. Our previous work [3] outlined the top-level architecture of such an authentication and key distribution protocol, and was motivated by the lack of security features in the architectural guidelines from ETSI standardizations, IETF, and academic research [3], since no protocol was found supporting the (1) authentication of VNFs, (2) negotiation of keys and (3) dynamic setup of secure VPN connections between VNFs. However, the principles of such requirements are similar to the Generic Bootstrapping Architecture (from 3GPP) [18] and Kerberos [19]. The main difference to this research problem is that it is the cryptoVNF and not the end-user that is involved in the authentication process.

In networks controlled and provisioned by operators, it is common to use a management plane or a control plane to provision the network topology. Therefore it is also possible to run the authentication process in a separate control plane domain. This is similar to the separation of the control and data plane in SDN where SD-IKE [20] is commonly utilised. However, SD-IKE does not describe how to securely distribute the keys between the network controller and the VNF, neither how the VNFs can preserve SFC packet headers during encryption, nor how it performs compared to regular IPsec setups.

This lack of SFC header preservation is also reflected through our previous work [2], showing that in contrast to the data, the SFC headers cannot be encrypted in order to enable the routers to perform SFC routing of the encrypted data. Therefore, the SFC header must be located between layer 2 and layer 3. Hence, encryption of layer 2.5 or layer 3 is needed in order to not interfere with the end-user data, where layer 3 encryption by the use of IPsec [21] in transport mode is preferred. IKE [22] and IKEv2 [23] are the main protocols used for key negotiations for IPsec, but currently, they require modifications in order to support a dynamic NFV environment with IKE over NSH.

Due to the backbone network in the examined scenarios (Figure: 2), encryption should ultimately be performed on the NSH layer. However, encryption on the NSH layer has currently no standard, except for securing the integrity of the NSH headers [24]. Furthermore, encryption on the upper transport layer by the use of SSL/TLS such as OpenVPN [25] is also possible. However, the end-to-end transport layer between VNFs is distinct from the end-user transport layer, creating additional overhead and potential packet segmentation or delay. Also, TLS based tunnelling is often based on endpoint attributes, such as an URL identity, something that does not fit to a site-to-site VPN setup. Similarly, the Wireguard [26] protocol that is an alternative to OpenVPN and IPsec, also simply encapsulates encrypted packets in a UDP header. Yet, Wireguard and OpenVPN have no good solution for key distribution and key derivation, as one key pair is used in the long-term and for all communication [27].

Furthermore, IETF proposed an Interface to Network Security Function (I2NSF) [28], to enable the exchange of secured messages between the VNFs and a security controller. This approach focuses on an out-of-band interface to operate the VNF, but this interface lacks security functions for authenticating and validating the VNFs. Hence, another key feature of our adversary model, is the characteristics of a separate control and management plane in NFV. This article adopts elements from authentication and key distribution in related autonomous control plane backbone networks [29], [30] to an NFV environment. Also, principles from distributed authentication protocols in Machine-to-Machine networks [31] and ad-hoc networks [32] [33] can be adapted to the NFV domain. Accordingly, the requirements both for authentication and key negotiation are defined in Section 2, in order to classify how the existing authentication protocols match the corresponding operational requirements.

Other related studies can also be identified in the literature, including Dynamic VPN architectures [34], Distributed VPN systems over peer-to-peer networks [35] [36] and security protocols for distributed systems [37]. All these articles refer to similar problems, but within distinct application domains, and although relevant, the proposed solutions are not directly applicable to the specifics of federated NFV environments.

2. Extraction and Discussion of Requirements

The core requirement for the examined scenario is a centralised system that can dynamically configure pairwise VPN channels between VNFs. IPsec is the most common approach for supporting encryption, wherein other environments, the two parties have a preshared key or a set of PKIs that is used to negotiate an encryption key by the use of the IKE protocol. IPsec is also selected as the encryption protocol for NFV. Yet, a mechanism is required in order to support the dynamic setup of IPsec channels, since the nature of VNFs in an NFV environment differs from normal endpoints in an IPsec channel. These differences are based on the following constraints of the NFV environment:

- It is the encrypting service function and not the end-user that act as VPN clients. Hence, the VPN setup is perceived as a site-to-site VPN setup, where the sites (or gateways in this case) are VNFs with encryption capabilities. From an end-user perspective, it should be of no concern how the dynamic VPN is provided, because neither the end-user nor the end-user device are participating in the process.
- The VNFs dynamically change their connection topology according to the specified SFC. At one point in time, VNF A is connected to VNF B, while at another point in time VNF A is connected to VNF C (Figure: 3). Both a network failure or a user-initiated request can trigger such a change in the service chain, at which time a centralised service such as an Authentication Centre (AuC) must inform the VPN gateways to reconfigure the VPN channels and derive new encryption keys.

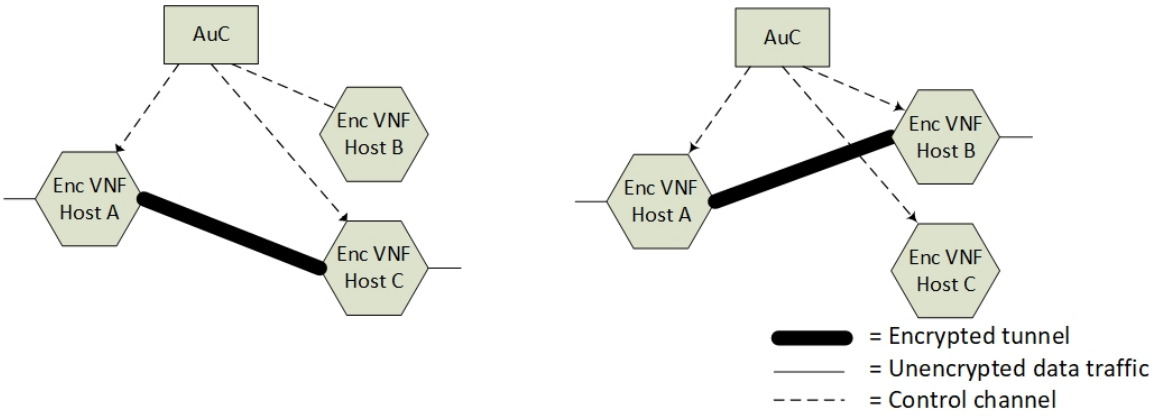

Figure 3: The dynamic behaviour of VPN tunnels

- It is the orchestration system (OSS) that defines the service chain and therefore controls the topology. Hence, the OSS must distribute VPN tunnel properties to the VNFs (which are acting as VPN peers). Therefore, the setup of the VPN peers must be initiated from a centralised unit such as an SDN controller or the OSS.
- In NFV environments, VNFs are enabled by containers or virtual machines, which operationally have a slow startup (especially a virtual machine). Therefore, it is not preferred to boot up VNFs on demand every time a service chain changes. Hence, the encryption functions must be pre-initiated for every compute node that contributes in an SFC. Also, the encrypting service function must have a secure channel to the Authentication Centre (AuC).
- A VNF application running encryption does not have to consider routing. The underlying NSH/MPLS layer is ensuring that the packet is routed correctly. Hence, the VNF encryption application can route all traffic through the application.

These constraints and functional requirements result in a set of requirements specific to the processes of authentication and key negotiation.

- **A trusted third party:** The involved parties must be authenticated (VNF-VNF and VNF-AUC), preferably with the use of a dedicated authentication server.
- The authentication and connection properties are dependent on the SFC. Hence, **context-based authentication** is required, in order to determine the physical location and SFC belonging of the VNFs.
- The AuC must be able to authorise service requests after authentication. This implies **distinguishing and isolating service requests** from different clients. For example, if a session between the VNFs and the AuC ensures confidentiality for transporting the service request, the AuC application also has to associate the incoming service request with this session. This ensures that a VNF cannot inject service requests for another VNF.
- Due to the SFC transport between VNFs, a dedicated control channel, typically provided by the backbone network, must be used for the authentication protocol. This control channel is preferable since the setup of VPN tunnels and backup tunnels should

not be dependent on the NSH/MPLS tunnel, while this communication must be **carried by the IP**.

- It is the VNF that has to initiate the authentication process since it must announce its presence. However, it is the AuC that must initiate the setup of an IPsec channel between two VNFs, since the AuC is the party that knows which VNFs that are required by the SFC. For that reason, the authentication protocol needs to support **both server-side and client-side initiated authentication**.
- Each VNF must be uniquely identified, with the identifier being pre-provisioned from the orchestration system to both the AuC and the VNF. **The VNF identifier** and the key are considered equivalent to a username and password pair.
- Due to the dynamic topology of the VPN channels, the corresponding keys cannot be static. This implies that in addition to the pre-provisioned static keys, additional keys for tunnel setup between the VNFs must be dynamically derived. After an initial authentication between the VNF and the AuC, these derived keys must be transferred securely to the VNFs. However, the initial keys used in the initial authentication must also be protected from eavesdropping. Hence, tickets or random numbers are needed in order to protect these credentials with the use of **confidentiality mechanisms during key derivation.**
- When a VNF is authenticated by a third party, the protocol must supply the VNF with a **remote connection gateway** for VPN setup.

Table 1 maps these requirements to the properties and supported functions of existing authentication protocols.

Point to Point Protocols (PPP) such as PAP [38], CHAP [39] and MSCHAP [40] are mainly designed for link layer authentication and do not focus on key distribution from a third party. These methods are also often used in AAA protocols such as RADIUS[41] and TACACS[42], reflecting point-to-point client-server authentication. Protocol overlay frameworks, such as EAP[44], have been developed to use an underlying protocol to carry the EAP messages. This is also typically designed to be used when IP is not available and there is a need for carrying authentication messages over multiple link-layer hops. An authentication method connected to layer 3 or layer 2.5 encryption is the main objective of this article. IPsec authentication variants have different methods for authenticating two peers, such as IKEv1 [22], IKEv2 [23] or KINK [56]. However, they all assume that the peers know the address of a remote peer before the authentication method starts, accordingly necessitating their extension in order to fully satisfy the aforementioned requirements.

Hop-by-hop tunnelling can also be achieved by enabling tunnelling on the transport or the session layer (layer 4/5). However, these hop-by-hop tunnels must not be mixed layer 4-7 data from the end-user such as SSL/TLS [47] layers. This implies that such tunnels must be implemented as underlying hierarchical tunnels where IP is transported over a layer 4-7 tunnel, which would require additional underlying hop-by-hop IP tunnels and an overlying orchestrating application. This additional overhead makes such tunnels non-preferable. Furthermore, application-based authentication relies on authentication messages that are encapsulated by application plane markup languages such by XML or REST messages. Examples of such protocols are OpenID and SAML, while these messages often rely on an

	Third party auth	Context-based auth.	Service request isol.	Supports IP transp.	Serverside auth	VNF identifier	Key derivation conf.	Remote connection attr.	
PPP protocols	✗	✗	✗	✓	✗	✓	✗	✗	PAP[38], CHAP[39], MSCHAP[40]
AAA protocols	✗	✓	✗	✓	✗	✓	✗	✗	RADIUS[41], TACACS[42], DIAMETER[43]
Protocol overlays	✗	✓	✓	✓	✓	✓	✓	✗	EAP[44], PEAP[45], PANA[45], LEAP[37]
IP layer	✗	✗	✓	✓	✓	✗	✗	✗	IKE-P1[22], IKEV2-AUTH[23]
Transport and session layer	✗	✗	✓	✓	✓	✗	✗	✗	TLS[46][47], DTLS[48]
Security applications	✓	✗	✓	✓	✗	✓	✗	✗	SAML [49], OAuth [50]
Generic bootstrapping	✓	✓	✓	✓	✓	✗	✗	✗	EAP-AKA[18]
Orchestration appl.	✓	✓	✓	✓	✓	✓	✗	✗	DMVPN[51], EAP-KMS[52], GSAKMP[53]
SW Security framworks	✗	✗	✗	✗	✗	✗	✗	✗	SASL [54], GSS-API [55]
Key Management Systems	✓	✓	✓	✓	✗	✓	✓	✗	KERBEROS[19], KINK[56]

Table 1: Authentication protocol

end-to-end transport mechanism, such as TLS, in order to ensure the confidentiality and integrity of the messages. This does not resolve the underlying identification and authentication problem where the VNFs do not know the remote endpoint and a URL does not exist.

Another approach for orchestrating authentication is to distribute network configuration through a network orchestrator, which is a common approach for many network vendors. Cisco uses for instance Group Encrypted Transport (GET) [57] to distribute VPN configurations from a server down to the clients, with multiple similar protocols and standards been suggested for the distribution of such configurations [58],[51], [59]. Yet, these solutions only distribute the initial keys, without having the capability of changing the configuration of the VPN topology rapidly and dynamically. Furthermore, other IPsec extensions distribute keys more efficiently [53],[60], but they can neither distribute information about the endpoints that need to be connected, nor rely on a third party being responsible for endpoint configuration.

The dynamic key distribution is one of the most critical features in designing automation of VPN setup for VNFs. A relevant approach is to use a dynamic key distribution protocol such as Key Management Systems that includes two-sided authentication such as Kerberos [19] or GPAKE [61]. However, Kerberos has security properties which impede the orchestration in a multi-domain environment [62], especially in cases where the remote VPN

peer must be received dynamically. Also, GPAKE has similar restrictions and does not have the capability of uniquely identifying a VNF identifier. On the other hand, Kerberos has an IPsec authentication extension named KINK [56] that makes it suitable for combining it with IPsec with service-side authentication. However, the protocol only supports IKEv1 and it has no distribution of remote endpoints. Accordingly, no existing protocol was found to fully satisfy the aforementioned requirements. Yet it was identified that a suitable solution would be a framework around IPsec, that also enables a fast, dynamic, and flexible key distribution. Furthermore, from the perspective of an NFV operator, an orchestrated solution fitting into the NFV framework is preferable, such as an API based architecture with principles from RESTconf [63].

3. Architecture

Based on these results, an authentication protocol and a key distribution mechanism are suggested. Figure 4 explains the top-level operation of the protocol derived from the aforementioned constraints, within a simplified scenario with 2 VNFs and an Authentication Centre (AuC). The simplified process consists of VNF instantiation, VNF authentication and VNF Configuration.

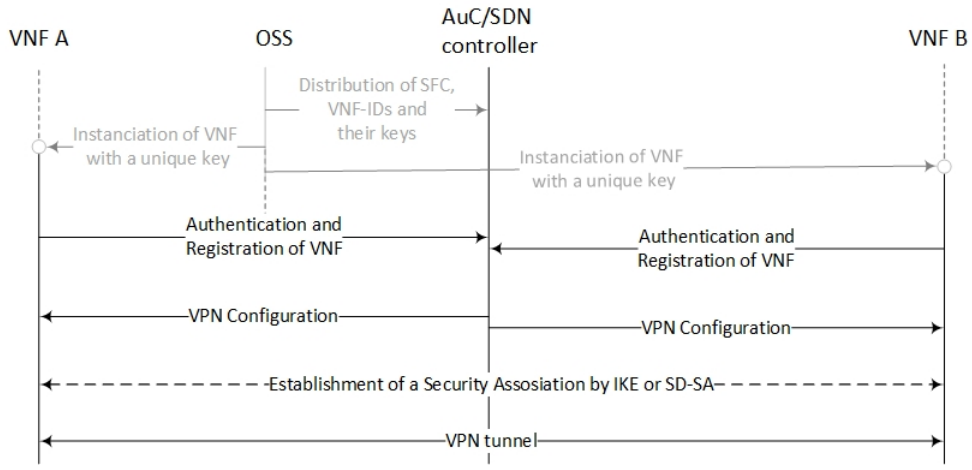

Figure 4: Simplified operation

We have based the proposed architecture on the principles of IPsec and RESTconf, building the framework around RESTconf in order to enable it to configure IPsec running inside VNFs in a dynamic manner. Our main contribution is therefore an architecture that automates the setup of IPsec over RESTconf where the IPsec services are running inside a VNF in an NFV environment. The proposed framework consists of three main components: (1) The VNF with encryption capabilities (VNF aka cryptoVNF), (2) an Authentication centre (AuC) and (3) an SDN controller, while all the components communicate by web services using the JSON format [63]. Figure 5 shows the bootstrap sequence and the communication between the different components. The bootstrapped mechanism consists of five steps:

0 Creation of VNFs, pre-distribution of keys, and definition of an SFC.
1 Mutual Authentication between the VNFs and the AuC.
2 Setting up configuration channels (RESTconf) between the VNF and the AuC.
3 Distribution of VPN configuration to the VNFs
4 Tunnel setup Local application.

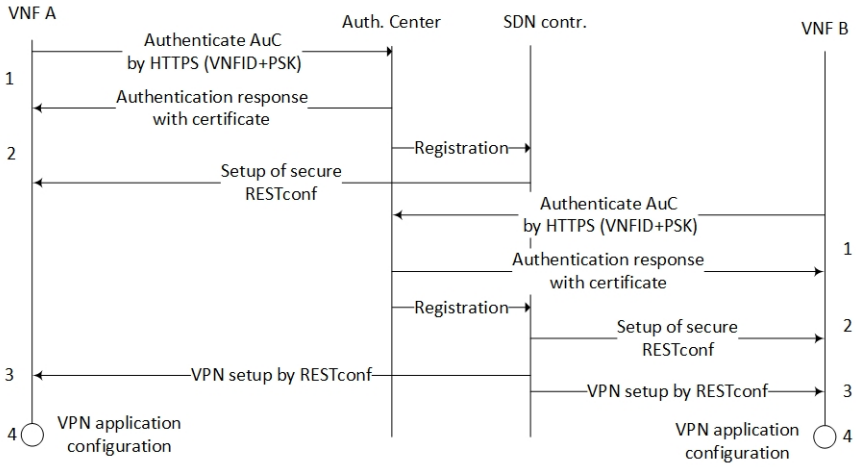

Figure 5: Detailed sequence diagram

0 **Distribution of VNFs with encryption capabilities.** We assume that an orchestration system is requested to set up a service chain and accordingly calculates the number of VNFs and encrypted channels. The VNF manager could then instantiate the VNFs, and pass them a globally unique VNF identifier and a preshared key. In a VMware environment, these can be set as parameters in the VMX file, enabling the guest OS to retrieve this information from the hypervisor during bootup. We skipped this initial step from the implementation of the architecture (numbered as 0), since the orchestration system functionality is not the focus of our contribution, while for proof of concept purposes the identifier and key were manually encoded into the machine.id parameter in the VMX file together with the hardcoded address of the AuC.

1 **Authentication and registration of an encryption VNF service.** After the VNF Manager has booted the VNF services, a registration service is initiated for the VNF, collecting the VNF identifier and the preshared key from the VMX file. Accordingly, the same service authenticates itself towards an AuC web service, where the authentication is ensured by SSL (HTTPS) with a certificate connected with the domain name. After a successful registration, the AuC pushes the identifier and the IP address of the VNF to a database of authenticated VNFs running on the SDN controller, while for every DHCP change the VNF reauthenticates.

2 **Setting up RESTconf** The second phase in the initialisation process is to establish a secure connection from the SDN controller to the VNF, in order to enable secure RESTconf messages. Because of the process in step 1, the SDN controller now has the IP address and the certificate to establish the TLS enabled RESTconf connection.

3 **RESTconf configuration pushing.** An SDN controller application is configured to push VPN configuration down the VNFs when all VNFs have booted and registered. The southbound interface of the SDN controller uses RESTconf to send a set of standardised configuration settings defined by NETconf YANG [64].

4 **IPsec application setup** The last step in the process is to configure the VPN application, which parses the NETconf YANG configuration into the application specific configuration settings.

This mechanism also allows the encryption application running inside the VNF to be unaware of the remote VPN peer when it is instantiated. Furthermore, the proposed mechanism also enables the use of Software Defined IKE (SD-IKE), because the two peers in the VPN are already authenticated towards an SDN controller, IPSec IKE becomes redundant. Accordingly, it is possible for the SDN controller to distribute the keys instead of IKE negotiations. Hence our second contribution in this paper is an architecture that enables automation of the setup of SD-IKE. The current SD-IKE draft suggests to provision this over I2NSF, which has limited features in a multivendor setup. Therefore, here we present an architecture for automating the VNF setup by the use of RESTconf and standard VNFs. We call the proposed approach Software Defined Security Associations (SD-SA), which is based on provisioning standard IPsec without IKE but over RESTconf.

Figures 6 and 7 show the differences between running IPsec with (existing approach) and without (suggested approach) the IKE protocol. The main objective of the IKE protocol is to authenticate the peers in order to populate the Peer Authorization Database (PAD) and distribute symmetric keys by populating the Security Policy Database (SPD) and Security Association Database (SAD). Within the mechanism presented earlier (Figure 5), the peers are already authenticated and the centralised controller is capable of replacing the IKE protocol. Instead of requiring IKE to populate the kernel databases, the SDN controller is populating directly the SAD and SPD databases, in order to reduce IKE resource consumption on the peers. In normal IKE setups, a Security Association is established between the peers, where the keys are transmitted. When not running IKE, the SA is not established between the peers, but the keys are distributed by the SDN controller. Since a secure channel exists for VPNGW1-SDNContr and VPNGW2-SDNContr (Figure: 7), the sum of these two channels is perceived as the aforementioned Software Defined Security Association (SD-SA).

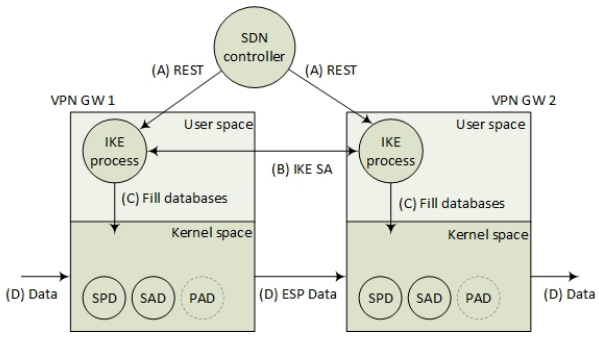

Figure 6: Updating IPsec configuration by RESTconf and making a Security Association (SA) by using IKE

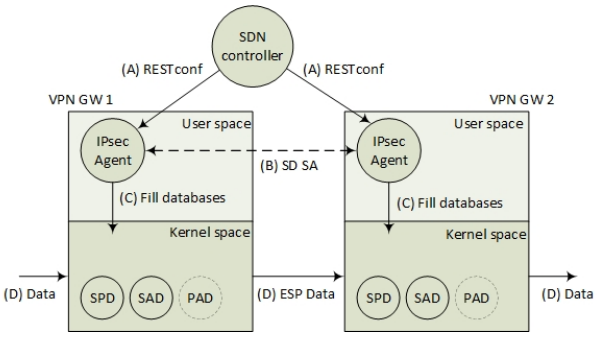

Figure 7: Distributing keys directly from controller by RESTconf and making a Software Defined Security Association (SD-SA)

In these two cases, the RESTconf API must support the distribution of 1 - RESTconf based IKE configuration and 2 - distribution of PAD+SAD configurations of RESTconf (SD-SA). In the first case, a REST message is sent with basic IKE preshared keys and connection settings, while in the second case the REST messages contain the same symmetric integrity key and encryption key.

4. Implementation

The main objectives of the implementation were to present an instance of the proposed mechanisms, to perform a proof of concept test and a performance comparison between IPsec/IKE and IPsec/SD-SA. Figure 8 shows the four main components in the software design:

1. The registration service on the VPN peer
2. The SDN controller acting as a configuration client and AuC
3. The local configuration service on the VPN peer
4. The IPsec service

Furthermore, Figure 9 provides a simplified flowchart and pseudo-code based description of the required processes.

0 **The VNFManager** is omitted for the implementation. That means that we manually defined the VPN pairs in the AuC. We also created an ISO image template in VMware and hardcoded the VNF-ID and the preshared key during the instantiating of the VNFs (in this case virtual machines).

1 **The registration client** is running in the VPN peer (the VNF) ensuring that a secure channel is set up between the controller and the VPN peer. On the VPN peer, the guest operating system application package OpenVMtools provides an API to get the VNF-ID and the PSK from the hypervisor. The application sends these credentials as an HTTP request to the AuC. This was implemented as a Linux bash shell script running wget commands. The service also updates the controller with the management address of the VPN peer.

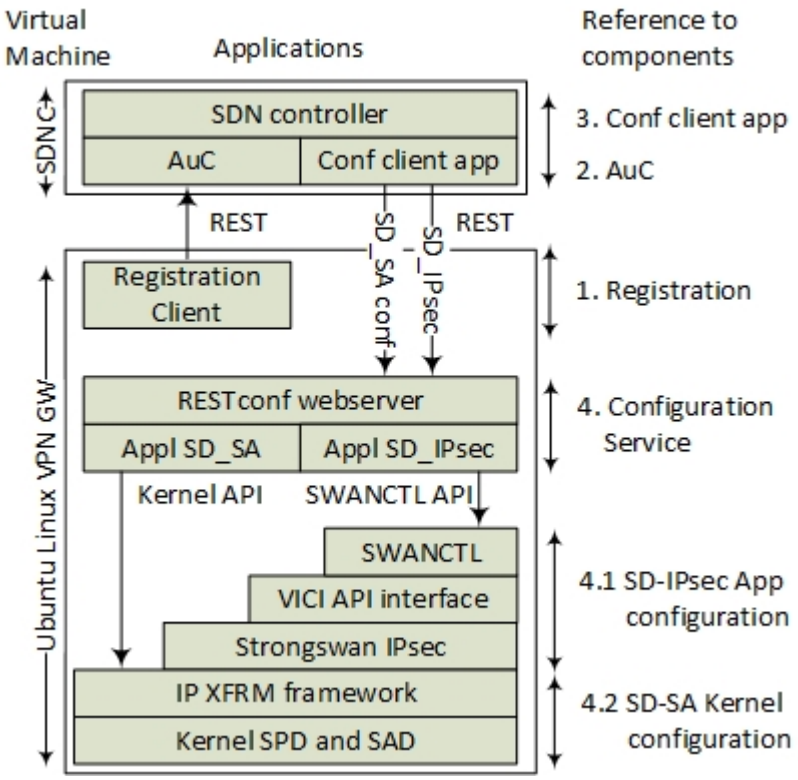

Figure 8: Components in the architecture

2 **The Authentication centre** is implemented as a web service running one authentication service. In the proof of concept experiment, the database is implemented as a simple text file containing all registered VNFs, their IDs, their PSKs and their management IP addresses. The web service authenticates the registration client by its ID and PSK, while the registration client authenticates the AuC by a certificate over a standard HTTPS connection.

3 **The Configuration client** runs a script that reads two sets of text file databases. The authenticated VPN peers and the set of defined VPN pairs. When a set of VPN peers is defined and both peers are registered, it sends the corresponding VPN configurations to the VPN peers. The configuration application reads the databases periodically and updates the VPN peers (the VNFs) with their configuration by RESTconf. Two versions of configuration options are enabled. The SD-IKE configuration configures IKE (Figure: 10), while the SD-SA configuration sets up an SA without the use of IKE (Figure: 11). The format of the configuration is based on the experimental updates of the expired IETF IPsec YANG specification draft [64]. For experimental purposes, we used a pseudo-JSON yang format that only contained a subset of the most important configuration parameters. For SD-IKEc the remote gateway, the identifier, the key lifetime and a preshared key were the most important parameters for making our proof of concept. For the SD-SA we defined 4 basic SDP policies per connection containing

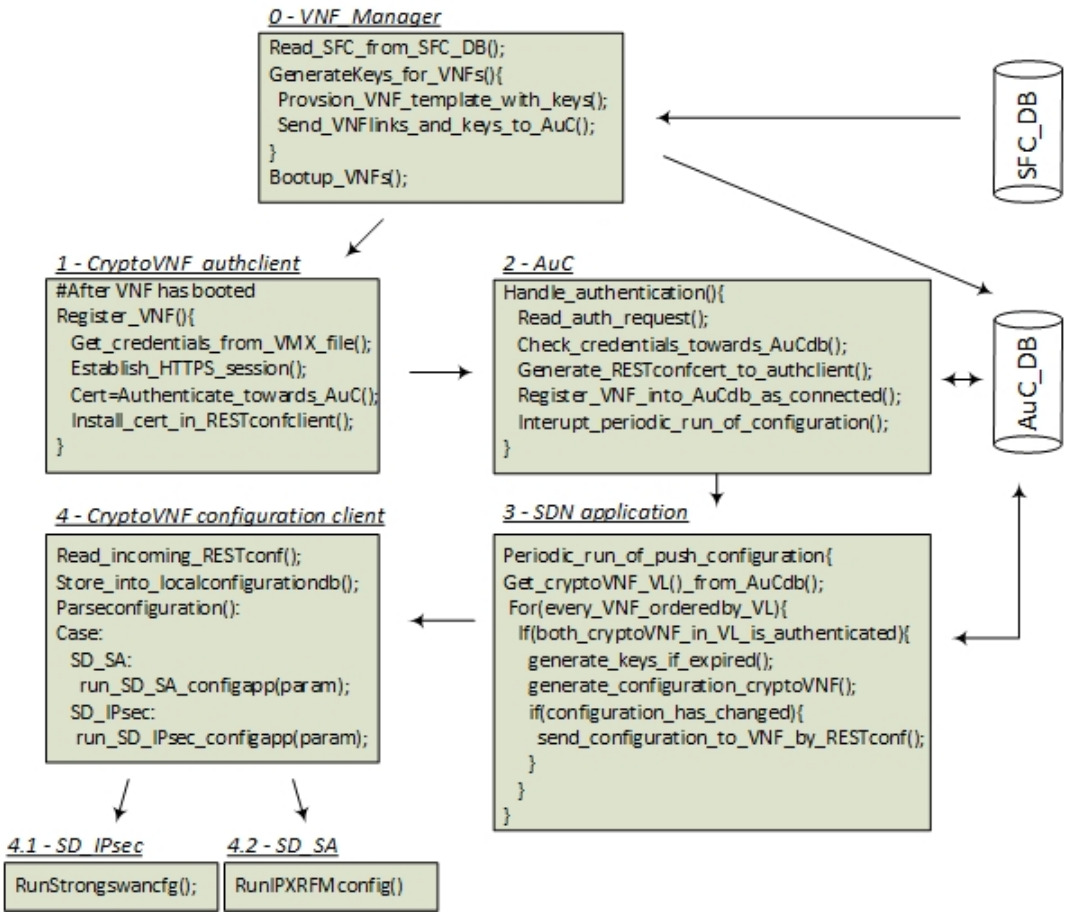

Figure 9: Flow chart and pseudo-code description of the processes

the connection id (ReqID) and the relevant IP address endpoints. Correspondingly, the SD-SA SAD state configuration contained only the relevant IP addresses of the endpoints, the IPsec header tags (SPI), the connection IDs (ReqID), the integrity keys and the encryption keys.

4 **The configuration service** is the local configuration server that is running on each VPN peer (VNF). The main objective for this service is to receive the VPN configuration from the centralised configuration server. This is implemented as an SDN controller that stores the VNF specific configuration locally. The application is storing the incoming configuration in XML files based on pseudo-RESTconf/YANG. We also simulated the notification service in NETconf by letting the service request trigger on-demand configuration changes to increase the deployment speed of the configuration. That was reflected by two different configuration web services that update the configuration. One application (4.1) configures the IPsec application with IKE configuration (SD-IKE) and another application (4.2) updates the kernel directly with the IPsec keys (SD-SA). Figure 13 and Figure 12 shows examples of the two southbound scripts that are configuring IPsec.

Figure 10: RESTconf YANG JSON data IPsec configuration (Step 3 to 4)

Figure 11: RESTconf YANG JSON data SD-SA configuration (Step 3 to 4)

4.1-4.2 **IPsec application** - We used Strongswan as the IPsec IKE application. The advantages with this application are that it supports both manual IKE configuration in text files, and that it has an API for controlling the IPsec configuration on demand. The Strongswan IPsec application has a dynamic library that enables such an interface.

However, we utilised the swanctl application overlay in order to enable Linux bash scripting CLI commands for updating the IPsec configuration without restarting the service. For the SD-SA application that manipulates the kernel IPsec configuration, we utilised the IP XFRM framework (Figure 13).

```
#script example manipulating IPsec configuration

#!/bin/bash
./generateconfigfromYang.sh > /usr/local/etc/ipsec.conf
ipsec rereadall;
for (( f=0; f<=$ConNb; f++ ))
do
 changeconfig = ´./getconfigstatus.sh $ConNb´
 if [ $ConNb -eq changeconfig ]
 then
  swanctl --terminate --ike $connections;
  swanctl --initiate --child ConNb$connections
 fi
done
```

Figure 12: Code example IKE configuration application (Step 4 to 4.1)

```
#Script example updating IPsec in the kernel with IP XFRM

ip xfrm state add src $IPsrc dst $IPdst proto esp spi 0x53fa0fdd
mode transport mark 0x$LblMark reqid $ReqID replay-window 32
auth "hmac(sha1)" 0x$Key1 enc "cbc(aes)" 0x$Key2

ip xfrm policy add dir out mark 0x$LblMark src $rcNet dst $Dstnet
ptype main action allow priority 2080 tmpl src $IPdst dst $IPsrc
proto esp reqid $ReqID mode transport

.....
```

Figure 13: Code example IP XFRM config application (Step 4 to 4.2)

5. Verification by experiments

The implementation was tested in order to run a proof of concept for the authentication and IPsec deployment mechanism, and to make a comparison of performance between IPsec/SD-IKE and IPsec/SD-SA. The performance test is primarily conducted in order to measure how much time it takes to change the data plane keys and how much overhead the key exchange protocol introduces with respect to resource consumption. Changing the data plane keys introduces a packet loss during rekeying. In IKE version 2, the specification

states that a new child SA should be established before the existing child SA is deleted. However, our measurements show that, even in this case, the Strongswan implementation still loses packets during rekeying. By sending a fixed stream of UDP packets through the VPN tunnel, the time-gap between rekeying the two peers is calculated (Formula: 1).

$$\text{Time gap} = \frac{\text{packets lost per key-change}}{\text{packets sent per second}} \tag{1}$$

In IKEv2 rekeying contains two major components. These are 1- Rekeying and reauthentication of the IKE SA session (aka Phase1 in IKEv1) and 2- Rekeying of the child SA (aka Phase2 in IKEv1) that contains the encryption and integrity key for the data plane. Usually, IKE SA rekeying is initiated every 3 hours, while child SA once every hour. In SD-SA, the IKE session is not established, while the rekeying is pushed by configuration changes from the controller. Hence, both IKE SA and child SA rekeying is compared with rekeying of SD-SA. Additionally, a key element in the SD-SA design is to compare the reestablishment of IKE. The requirement of the dynamic behaviour of reconnecting IPsec VNFs during a service chain modification (Figure: 3) is a feature that IKE is not designed to handle. However, SD-SA does not handle such configuration differences differently than normal rekeying. In our experiment, we have omitted routing table changes in such setups and we have also assumed that routing and key distribution is performed in one operation. Also, all IKE experiments are performed with the IKE version 2 since this version is known to be faster than IKE version 1 [65].

The requirement section (Section: 2) stated that the encryption functions are considered being pre-instanciated. In an NFV context, this implies the encrypting functions are available from a pool of specifically assigned and pre-instanciated VNFs outside of the regular VNF application domain. The deployment time of using an encryption function is therefore considered as the time it takes to establish a new VPN configuration, by sending a message from the AuC to the CrytoVNF. Hence, we do not measure deployment time and failover time, but we perceive the equivalent as IKE reconnection and SD-SA reconnection times.

Based on these IKE attributes we created five test scenarios in our experiment. Three IKE scenarios and two SD-SA scenarios.

- A reference performance test for IKEv2 is defined when no key-change takes place. This ensures that there is no packet loss without key-changes and defines the maximum bandwidth throughput
- A reference performance test for SD-SA was also defined with no key-change. This test case also verifies that there is no packet loss without key-changes.
- An SD-SA proof of concept implementation was tested to measure the resource consumption, the delay and packet loss during key-changes, and a bandwidth test measure the overall performance of the system.
- Running IKEv2 with child SA key-change is defined to measure and compare the SD-SA with the regular IKE key-changes.
- Running IKEv2 with IKE SA key-changes is defined to measure and compare these types of key-changes to SD-SA.

- The last test was IKE reconnections, to simulate VNF changes and compare it SD-SA reconnections.

As mentioned earlier, IKE normally uses 3600 seconds as a default interval value for rekeying. In order to calculate the time gap between the host during key-changes, we ran the tests with high key-change rates. These tests are not relevant in normal IPsec setups, but allow capturing the required measurements that ensure reliable results. We ran tests with 2- and 4-second rekeying interval for packet loss measuring, while for resource consumption measurements we used 4 and 30 seconds rekeying interval.

The test was performed in a VMware lab environment provided by Eidsiva broadband in Norway. 6 ESXi host based on HP Proliant DL360G9 with 24 CPUs x 2.6 GHz and 8 Gigabit Ethernet ports. 6 virtual machines were created, one per ESXi host. Each virtual machine was allocated 4 virtual CPUs and 8 GB of RAM. All hosts were installed with Ubuntu server 16.04 LTS and were running kernel 4.4.0-116 SMP. All servers were installed with standard installation settings with no kernel modifications. In order to reduce the number of unknown variables in the virtual machines, no additional services were installed. Additionally, the resources were reserved to the virtual machines and no other virtual machines were running on each ESXi host to ensure no resource sharing.

Figure 14 shows how two test agents were transmitting packets through a site-to-site VPN topology. The router in the middle had no other purpose than ensuring that non-encrypted traffic was able to pass the router. Each Virtual machine was interconnected with dedicated network 1 Gbps Ethernet interfaces, while the connections to the SDN controller were separated from the data plane interfaces.

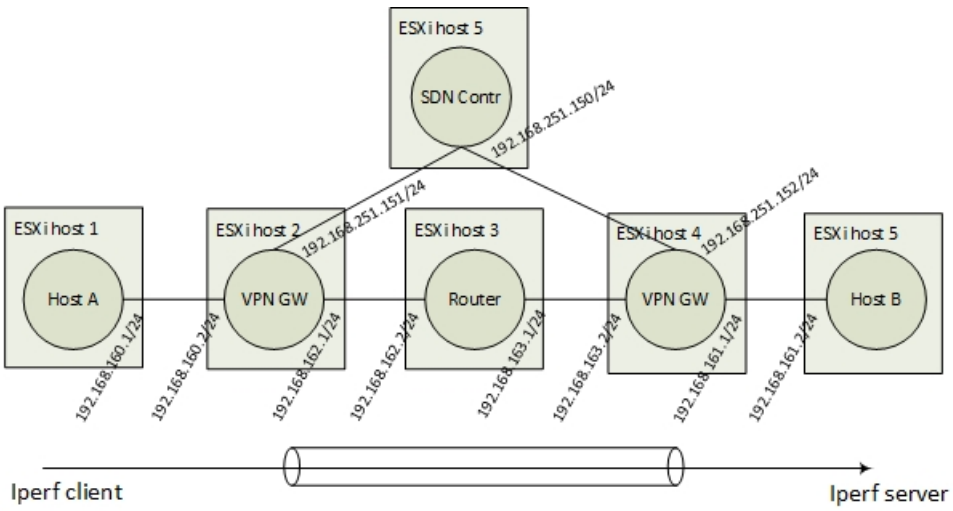

Figure 14: Lab topology

The VPN gateways were running Strongswan 5.5.0 git commit 8eea280, while the test agents where running iperf 5.0.2. The web services were simulated by the use of the socat application. This application is similar to netcat and capable of creating TCP/IP sockets

with SSL support, and was used to send pseudo REST commands based on the Linux bash scripting that was mentioned in the implementation section 4.

We used iperf as the testing tool for packet loss and bandwidth tests, performing bandwidth tests with a window size of 416K in 20 seconds. For UDP packet loss tests we ran a UDP stream of 100Mbit with packet size 578 bytes at 173100 pps. Additionally, a 50Mbit test with packet size 578 at 85072 pps was performed. Each test was performed 10 times where the result is an average this. For CPU resource consumption we took periodic measurements of the CPU usage by the nmon tool, with a total of 60 samples in discrete time. Finally, to measure the memory consumption we used the fstab tool to perform 60 discrete time measurements of the memory consumption. The results of the measurements are shown in Table: 2 and Table: 3.

		Settings		Bandwidth	100 Mbit UDP			50 Mbit UDP			Timegap
		Key change rate in seconds	Number of connections	TCP bandwidth Mbps Windowsize 416K Average of 10 tests	Average packets lost of 173100 pps 10 tests x 20 seconds	Average packetloss per keychange	Average timegap in ms per keychange	Average packets lost of 85072 pps 10 test x 20 seconds	Average packetloss per keychange	Average timegap in ms per keychange	Average timegap in ms per keychange
SDSA	Reference test no keych.	3600	1	717,4	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	SD-SA with keychange	2	1	660,0	828,0	82,8	3,9	176,1	17,6	4,1	
		4	1	702,0	348,8	34,9	4,1	162,1	32,4	3,8	4,0
IKEv2 SAs	Reference test no keych.	3600	1	701,8	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	Child SA keychange	2	1	655,8	166,1	16,6	2,1	48,2	4,8	1,1	
		4	1	682,0	67,4	13,5	1,8	29,4	5,9	1,5	1,6
	IKE SA keychange	2	1	NA	NA	NA	NA	NA	NA	NA	
		4	1	651,0	157,5	31,5	4,1	58,7	11,7	3,1	3,6
	IKE reconnect	2	1	569,0	10878,0	1088,8	127,4	4633,0	463,3	108,8	
		4	1	609,0	4948,0	989,6	115,8	2388,0	477,6	112,2	116,5

Table 2: Measuring performance and packet loss per key-change

	Settings		Bandwidth	CPU		Kernel memory use		User memory use	
	Keychange rate in seconds	Number of connections	TCP bandwidth Mbps Windowsize 416K Average of 10 tests	Average CPU kernel use, 4 of 4 CPUs in 20 discrete time measures	Average CPU userspace use, 4 of 4 CPUs 20 discrete time measures	Average of total kernel memory use in bytes, 20 discrete time measures	Kernel memory use per connection in bytes	Average of total user memory use in kbytes, 20 discrete time measures	User memory use per connection in kbytes
SD SA	4	30	699	2,60%	1,00%	129k	4,3k	0	0
	30	1000	648	5,80%	2,50%	4361k	4,4k	0	0
IKEv2 w/child SA keychange	4	30	668	1,60%	1,20%	125k	7,2k	13502k	453k
	30	1000	610	4,30%	3,20%	7817k	7,8k	38284k	38k
IKEv2 w/IKE reconnect	4	30	613	1,10%	2,60%	144k	4,8k	4813k	160k
	30	1000	603	22,50%	31,00%	7603k	7,6k	62705k	63k

Table 3: Measuring scalability and resource consumption

Most of the measurements presented in tables 2 and 3 are statistical averages, where variables such as packet loss and CPU number of software interrupts can potentially affect

the outcome. Yet, the results were verified by completing the test two times with a resulting variance of no more than 5%.

6. Discussion and Evaluation

The implementation and tests of SD-SA verify that it is possible to run IPsec without IKE in software-defined environments such as NFV. Instead of exchanging keys directly between two peers, a third party configuration server can distribute the keys directly to the peers. This is archived through pairs of secure configuration channels that are established between the VPN peers and a configuration server and replace the IKE channel. Our proof of concept implementation has the RESTconf configuration interface both can set up IKE configurations and SD-SA configurations. Accordingly, the proposed mechanism satisfies the requirements for efficient and dynamic configuration of IPsec VPNs in NFV environments. In this section, we discuss our performance results, analyse the security of our new mechanism and evaluate interoperability concerns.

6.1. The performance results

Table 2 and 3 showed the result of our performance test. Here, we discuss these result with respect to key-changes, throughput, resource consumption, latency and general benefits of SDN.

IKE SA key-change: A normal IKEv2 SA key-change (Phase 1 in IKEv1) runs periodically every 3-4 hours. However, this experimental setup with short key-change intervals, intends to compare the performance of the different IPsec key-changing methods. The results show that the initial IKE setup and periodic key-changes in IKEv2 SA spends about the same time changing keys as with SD-SA (in average 3.6ms vs 4.0ms in Table: 2). However, IKE SA demands more computation resources than SD-SA, such as 31,0% vs 2,5% CPU time Table: 3)

A consequence of this is also seen when running IKE reconnections with a 2-second interval, where the whole process congests and measuring is not possible. This is because the process is not finished before a new key-change thread is initiated (Table: 2).

IKE child SA key-change: Traditional IKEv2 with child SA key-changes (Phase 2 in IKEv1) has less time period of packet loss than non-IKE with SD-SA (1.6ms versus 4.0ms in Table: 2). However, this difference is not reflected in the TCP bandwidth test, where SD-SA, in fact, performs better than IKEv2 (682Mbps vs 702Mbps in Table: 2). A child SA key-change is more frequent than an SA key-change, but for SD-SA there is no difference in these type key-changes. Hence, the performance results of child SAs are considered more important and relevant. However, in virtualised environments with dynamic resource allocation, it is also reasonable to have many IKE reconnections due to Virtual Machine migrations [66].

Throughput: Interestingly, the performance tests show that the throughput is not significantly affected by the number of key-changes (Figure: 2). By monitoring the process consumption, we discovered that our Ubuntu operating system automatically dedicated CPU resources to data plane packet-handling (processor one) and control plane key-changes

(processor two) respectively. The Linux kernel on the VPN peers spend most of the CPU time on handling software interrupts when running VPN data traffic through them. This causes one of the CPU to peak close to a 100% utilisation while performing packet encryption. The reduction of 168 bytes of available packet size is another reason for the system not to be able to archive 717 Mbps with no key-change.

Resource consumption: With respect to resource consumption, we measured memory and CPU time without running any data plane traffic. Table: 3 shows that IKEv2 reconnections require a significant amount of CPU time. Both 30 and 1000 connections were not able to finish within the time intervals of 4 and 30 seconds. Therefore these measurements are not precise with respect to memory consumption per connection. The main reason for this delay is the waiting time for network packets from the remote peer. Hence, this result indicates that SD SA is a much more efficient way to reconnect IPsec sessions.

The reason behind the significant differences in memory consumption across the tests is that the IPsec application Strongswan prepares and installs the next keys in advance. Therefore the memory consumption is doubled. In our simplified experiment, we did not consider SA overlapping during key-changes for SD-SA. However, it is possible to have two incoming ESP packet policies that ensure that the VPN peer receiver does not delete the old IPsec policy before a new one is active. We chose not to implement this feature for both IPsec/SD-IKE and IPsec/SD-SA. In respect to the measurements in the experiment, this parallelism feature would not enable a precise measurement of the time gap for key-change delays. However, it is assumed that overlapping IPSec policies would reduce the packet loss. The main difference between the IPsec based on IKE or SD-SA would concern the amount of resource consumption.

Latency: Another factor that influences the performance is the distance and latency between the VNFs. In the presented test scenario the virtual machines are provisioned in a closed environment with direct peerings between the hosts, that enables a very low latency between the hosts. If the SDN controller is placed outside of the data centre, or if the VPN peers exist in different data centres, this would increase the latency between the peers and consequently affect the delay between the key-changes on each VPN peer. However, It is expected that this delay is similar for IPsec with IKE and for SD-SA. This is because the SDN controller sends a similar key update to both the SD-SA application and the IKE application and therefore the difference should be equally linearly to the latency. Also, for this proof of concept demonstration, we primarily aim to show that SD SA works equally or better than IKE in order to further apply it to an NFV domain. For that reason, this latency is not highly relevant when comparing the methods.

Communication constraints: Concerning performance, IKE and SD-SA perform relatively similar, but SD-SA performs slightly better when the number of key-changes is high. We have shown that both IPsec enabled with IKE and with SD-SA can ensure isolation of Virtual Links in federated NFV environments. However, SD-SA consumes less resources and has the advantage of not being dependent on the transport protocol on the data plane. For example, if the data plane transport channel is based on NSH, direct communication between the VNFs is not possible over NSH. Hence, a separate control-channel is required in order to make the VPN peers (the VNFs) exchange keys. This implies that, regardless

of performance, IPsec with SD-SA is more suitable in federated NFV domains due to these communication constraints. However, for comparison reasons, we did not use NSH on the data plane in our performance test

Benefits of SDN: Our approach of separating IKE from packet encryption is similar to related results performed by Vajaranta et.al [67]. Their experiment was based on utilising OpenFlow to load-balance IPsec. Their results showed that, for high bandwidths in particular, OpenFlow enhances the IPsec availability and performance. Our experiment was based on NFV with a secure distribution of keys and not based on OpenFlow as a load-balancer. However, the distributed design and flow-based control of SDN for both experiments emphasise the scalability benefit in distributed environments. This confirms our results, but it also indicates that our design of secure key distribution is applicable to other application domains such as load-balancing of IPsec.

IKE drawbacks in Virtualised environments: The consequence of VM migration and dynamic resource allocation in virtual environments also favorises SD-SA in front of IKE. Having encryption services running as VNFs implies that the encryption services can migrate between hosts and change virtual machines along with SFC changes. Every VM migration would require an IKE reconnection, while SD-SA only requires a key update if we follow our suggested architecture [1]. This clearly distinguishes VPN setups in virtual infrastructure domains and hardware-based VPN networks. If an SDN controller is responsible for both distributing encryption keys and performing routing, it is expected that such topologies can reduce the failover time compared to traditional BGP routing [68] and IKE IPSec. We aim to test this in our future work (see Section: 6.4).

6.2. Security Analysis

In this Section, we analyse the security of our proposal. Due to the use of multiple protocols, components and communication planes, a formal method or a code analysis is difficult to archive in order to analyse the level of security. Also, the operational characteristics of the key exchange mechanism are not fully specified, that makes a formal verification difficult. However, we did analyse standard network protocol security features and resistance against well-known attacks according to basic security principles [69] such as confidentiality, integrity, availability.

Confidentiality: The objective of our security mechanism is to keep the encryption key and the integrity key for the SA in IPsec private. For IKEv2, the peers derive these keys between each other from parameters such as the preshared key, while in SD-SA the keys are sent directly to the peers from the controller. Both methods require that the configuration channel between the controller and the VPN peer is protected. In both scenarios, the configuration channel is protected by SSL. This means that the underlying keys are dependent on the integrity and confidentiality of the configuration channel. This implies both the protection of the network, such as the quality of the ciphering algorithm, but also the protection of software components. The system fully relies on the orchestrator in distributing the keys to the VNFs. A compromised VNF or a compromised orchestrator therefore breaks the security and enables the adversary to launch attacks using the keys obtained.

Both IKE v1 and IKE v2 focus on multiple iterations of key derivations to make sure that the encryption key and the integrity keys are kept confidential. The keys are not transferred between the peers such as we suggested for SD-SA. However, for future extensions, the SD-SA key transfers are also possible to extend with additional key derivations such as Diffie-Hellman [70]. In addition to key derivations in IKE, the keys are also kept protected inside the kernel and the IPsec application and not shown in a configuration file. This makes the encryption key and the integrity key less available in a system that is not fully compromised.

We have not investigated how the security mechanism can be protected from a compromised SDN controller, authentication centre, orchestration system or VNF. However, it is assumed that additional security features such as integrity attestation of software packages must co-exist with the presented key exchange mechanism.

Integrity: It is not investigated how replay attacks are possible over the configurations channels. However, the NETconf protocol states that the underlying transport protocol must handle such protection [71].

Scalability The number of controllers and the number of virtual machines can easily be adjusted in a virtualised environment, However, the computational resources needed for encryption and decryption is closely connected to how much data traffic the end-users are consuming. Sudden changes in behavioural patterns, such as viral videos, could potentially demand more computational power than available in the NFV domain. Virtual environments use shared resources and often overbooked services. In use cases where each user runs SFCs with multiple encrypted Virtual Links, the computational need for performing encryption is exponential to the bandwidth utilisation and number of Virtual Links.

Availability: Both the VPN peers and the SDN controller is vulnerable for Denial of Service attacks. Especially, DDoS towards VPN peers can result in amplified resource consumption as mentioned in the previous section. For DDoS towards the SDN controller, we assume that it runs in a federated control plane domain [1] separated for the data plane. Hence, the attack surface is considered relatively low. However, a DDoS attack on the data plane in multiple VPN channels will increase the CPU resource consumption for all types of encrypted topologies. Hence, this problem will affect both IPsec/IKE, IPsec/SD-SA or other underlying IPsec channels in an SFC with a similar amount of resource consumption.

Another aspect of availability is network attacks on the IKE UDP port 500. Half-open IKE connection is a resource consumption problem that comes from establishing too many IKE connection. Because the IKE protocol often runs over a network port available on the data plane, IKE is more vulnerable for such attacks than SD-SA. This is because SD-SA is suggested to run over a control plane network that is separated from the data plane.

Reliability: Our implementation did not take into account the aforementioned scenarios where the SDN controller or other components become unavailable. For example, if the controller becomes unavailable, it is important that the VPN peers do not use the cipher key after the lifetime expires. For such cases, the local configuration service in the VPN peer has to ensure that the key expires if no key is received from the SDN controller. We did neither consider system responses to deadlock or system crash in any of the components.

6.3. Interoperability

This article suggests a new key distribution paradigm for the encryption of Virtual Links per SFC in NFV. This raises an interoperability problem of both the VNFs and the NFV infrastructure components. From a NFV infrastructure perspective, the suggested architecture is proposed in interconnected and federated NFV domains. This implies that the underlying infrastructures support multi-tenant domains, where each tenant, in theory, is capable of running their own customer-specific SFC routing method, supported by a network overlay. However, VNF interoperability is more challenging. According to the SFC specification, the VNFs can be (1) SFC-aware or (2) SFC-unaware supported by an SFC aware proxy. This implies that the SFC proxy or the VNF must be aware of the SFC routing mechanism, such as the NSH headers. Consequently, the encrypting VNFs must also support SFC routing. The different SFC routing methods, such as segment routing in MPLS, IPv6 or NSH, put a burden on VNF developers in supporting different standards. This is a general VNF problem and not specific to our application. However, our application introduces an additional parameter for the VNF developer to consider. It also raises a new standardisation issue of encryption applications and their programming interface towards the AuC and how to deal with different types the SFC data forwarding standards. Hence, this proof of concept experiment aims to contribute to the standardisation of VNF application interfaces for enabling encrypted Virtual Links. This also indicates a need for a standardised encryption header in the SFC protocol, such as NSH.

6.4. Future work

In this article, we chose to focus on the security mechanism of the key-exchange between virtual encryption functions for providing isolated SFCs. We did not take SFC transport mechanisms such as MPLS or NSH into account when we performed our measurements. Neither did we consider topology changes and the effect of routing protocols delays in our design. For future work, the routing protocol of the SFCs needs to be aware of the cryptographic endpoints for every hop in an SFC. This brings an additional cryptographic attribute the NFV routing and resource allocation problem [72]. This optimization problem is an NP-hard problem that we aim to resolve by distributing the routing decisions by the use of multiprotocol BGP [68]. Consequently, our future work relies on providing a testbed for integrating the encryption functions and the routing mechanism into an NFV testbed.

7. Conclusion

We have presented a new way of utilising SDN in NFV by automating the key distribution in the setup of secure VPN channels between VNFs. This mechanism was specifically developed in order to enable per-flow encryption in federated NFV environments. However, it also solves the communication problem of establishing an IKE Security Association between two VNFs in an SFC. Through our proof of concept demonstration, we have shown that the automation procedure can be utilised to setup Security Associations between VPN peers for both IKE and non-IKE IPsec VPN connections. However, in comparison, the proposed SD-SA mechanism can be even more efficient and scalable than traditional IKE.

The results of the performance tests show that the delay between rekeying the VPN peers are slightly faster when running IKE, while SD-SA requires less resources. The presented architecture can be utilised both for small and large data centre deployments. However, the automated key distribution mechanism is excepted to have the greatest benefit in an NFV environment, with a lot of encrypted channels, such as in a per-flow per service chain encryption. The proposed bootstrapped mechanism is based on standard internet security protocols such as HTTPS and RESTconf where the majority of the security relies on these underlying protocols. We have seen that the most critical factor for our proposal is to have available compute resources for encryption and decryption.

Author Contributions:

The main author of this paper is H.G.; V.G. and T.K. have contributed with respect to the methodology, paper structure, quality assurance and editing.

Funding:

This research was funded by Eidsiva, the Norwegian Research Council and the Norwegian University of Science and Technology (NTNU).

Conflicts of Interest:

The authors declare no conflict of interest.

References

[1] H. Gunleifsen, V. Gkioulos, T. Kemmerich, A tiered control plane model for service function chaining isolation, Future Internet 10 (6) (2018) 46.

[2] H. Gunleifsen, T. Kemmerich, Security requirements for service function chaining isolation and encryption, in: 2017 IEEE 17th International Conference on Communication Technology (ICCT), pp. 1360–1365.

[3] H. Gunleifsen, T. Kemmerich, S. Petrovic, An end-to-end security model of inter-domain communication in network function virtualization, Norsk Informasjonssikkerhetskonferanse (NISK): Bergen, Norway (2016) 7–18.

[4] European Telecommunications Standards Institute (ETSI), Network functions virtualisation (nfv): Architectural framework v1.1.1Available online: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfvman001v010101p.pdf (accessed on 24 January 2019).

[5] N. Figueira, R. Krishnan, D. Lopez, S. Wright, D. Krishnaswamy, Policy Architecture and Framework for NFV Infrastructures, Active Internet-Draft, IETF Secretariat, Internet-Draft draft-irtf-nfvrg-nfv-policy-arch-01.

[6] J. M. Halpern, C. Pignataro, Service Function Chaining (SFC) Architecture, RFC 7665 (Oct. 2015). doi:10.17487/RFC7665.

[7] V. Manral, S. R. Hanna, Auto-Discovery VPN Problem Statement and Requirements, RFC 7018 (Sep. 2013). doi:10.17487/RFC7018.

[8] S. Hyun, J. P. Jeong, T. Roh, S. Wi, P. Jung-Soo, I2NSF Registration Interface Data Model, Internet-Draft draft-hyun-i2nsf-registration-interface-dm-06, Internet Engineering Task Force, work in Progress (Jul. 2018).

[9] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, R. Kumar, Framework for Interface to Network Security Functions, RFC 8329 (Feb. 2018). doi:10.17487/RFC8329.

[10] A. Atlas, J. M. Halpern, S. Hares, D. Ward, T. Nadeau, An Architecture for the Interface to the Routing System, RFC 7921 (Jun. 2016). doi:10.17487/RFC7921.

[11] P. Quinn, U. Elzur, C. Pignataro, Network Service Header (NSH), RFC 8300 (Jan. 2018). doi:10.17487/RFC8300.
[12] A. M. Alwakeel, A. K. Alnaim, E. B. Fernandez, A survey of network function virtualization security, in: SoutheastCon 2018, IEEE, 2018, pp. 1–8.
[13] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, H. Flinck, Network slicing and softwarization: A survey on principles, enabling technologies, and solutions, IEEE Communications Surveys & Tutorials 20 (3) (2018) 2429–2453.
[14] M. D. Firoozjaei, J. P. Jeong, H. Ko, H. Kim, Security challenges with network functions virtualization, Future Generation Computer Systems 67 (2017) 315–324.
[15] S. Lal, T. Taleb, A. Dutta, Nfv: Security threats and best practices, IEEE Communications Magazine 55 (8) (2017) 211–217.
[16] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. López, K. Papagiannaki, P. Rodriguez Rodriguez, P. Steenkiste, Multi-context tls (mctls): Enabling secure in-network functionality in tls, ACM SIGCOMM Computer Communication Review 45 (4) (2015) 199–212.
[17] K. Bhargavan, I. Boureanu, A. Delignat-Lavaud, P.-A. Fouque, C. Onete, A formal treatment of accountable proxying over tls, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 799–816.
[18] J. Arkko, V. Lehtovirta, P. Eronen, Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA), RFC 5448 (May 2009). doi:10.17487/RFC5448.
[19] D. C. Neuman, S. D. Hartman, K. Raeburn, T. Yu, The Kerberos Network Authentication Service (V5), RFC 4120 (Jul. 2005). doi:10.17487/RFC4120.
[20] R. Lopez, G. Lopez-Millan, Software-Defined Networking (SDN)-based IPsec Flow Protection, Internet-Draft draft-ietf-i2nsf-sdn-ipsec-flow-protection-02, Internet Engineering Task Force, work in Progress (Jul. 2018).
[21] K. Seo, S. Kent, Security Architecture for the Internet Protocol, RFC 4301 (Dec. 2005). doi:10.17487/RFC4301.
[22] P. E. Hoffman, Algorithms for Internet Key Exchange version 1 (IKEv1), RFC 4109 (May 2005). doi:10.17487/RFC4109.
[23] P. Eronen, Y. Nir, P. E. Hoffman, C. Kaufman, Internet Key Exchange Protocol Version 2 (IKEv2), RFC 5996 (Sep. 2010). doi:10.17487/RFC5996.
[24] T. Reddy, P. Patil, S. Fluhrer, P. Quinn, Authenticated and encrypted NSH service chains, Internet-Draft draft-reddy-sfc-nsh-encrypt-00, Internet Engineering Task Force, work in Progress (Apr. 2015).
[25] I. Kotuliak, P. Rybár, P. Trúchly, Performance comparison of ipsec and tls based vpn technologies, in: 2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA), IEEE, pp. 217–221.
[26] J. A. Donenfeld, Wireguard: next generation kernel network tunnel, in: 24th Annual Network and Distributed System Security Symposium, NDSS, 2017.
[27] A. S. Braadland, Key management for data plane encryption in sdn using wireguard, Master's thesis, NTNU (2017).
[28] S. Hyun, J. Kim, H. Kim, J. Jeong, S. Hares, L. Dunbar, A. Farrel, Interface to network security functions for cloud-based security services, IEEE Communications Magazine 56 (1) (2018) 171–178.
[29] K. M. Eie, Authentication in protected core networking, Master's thesis, NTNU (2016).
[30] N. Okabe, S. Sakane, K. Miyazawa, K. Kamada, M. Ishiyama, A. Inoue, H. Esaki, Implementing a secure autonomous bootstrap mechanism for control networks, IEICE Transactions on Information and Systems 89 (12) (2006) 2822–2830.
[31] J. Latvakoski, A. Iivari, P. Vitic, B. Jubeh, M. B. Alaya, T. Monteil, Y. Lopez, G. Talavera, J. Gonzalez, N. Granqvist, et al., A survey on m2m service networks, Computers 3 (4) (2014) 130–173.
[32] L. Zhou, Z. J. Haas, Securing ad hoc networks, IEEE network 13 (6) (1999) 24–30.
[33] N. Aboudagga, M. T. Refaei, M. Eltoweissy, L. A. DaSilva, J.-J. Quisquater, Authentication protocols for ad hoc networks: taxonomy and research issues, in: Proceedings of the 1st ACM international

workshop on Quality of service & security in wireless and mobile networks, ACM, 2005, pp. 96–104.
[34] Y. Gao, C. Phillips, L. He, Dvm based dynamic vpn architecture for group working and orchestrated distributed computing, in: Third International Conference on Digital Information Management, 2008. ICDIM 2008., IEEE, 2008, pp. 763–768.
[35] S. Gokhale, P. Dasgupta, Distributed authentication for peer-to-peer networks, in: 2003 Symposium on Applications and the Internet Workshops, IEEE, 2003, pp. 347–353.
[36] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, H. Tokuda, Ela: a fully distributed vpn system over peer-to-peer network, in: The 2005 Symposium on Applications and the Internet, IEEE, 2005, pp. 89–92.
[37] S. Zhu, S. Setia, S. Jajodia, Leap+: Efficient security mechanisms for large-scale distributed sensor networks, ACM Transactions on Sensor Networks (TOSN) 2 (4) (2006) 500–528.
[38] W. A. Simpson, PPP Authentication Protocols, RFC 1334 (Oct. 1992). doi:10.17487/RFC1334.
[39] W. A. Simpson, PPP Challenge Handshake Authentication Protocol (CHAP), RFC 1994 (Aug. 1996). doi:10.17487/RFC1994.
[40] G. Zorn, Microsoft PPP CHAP Extensions, Version 2, RFC 2759 (Jan. 2000). doi:10.17487/RFC2759.
[41] A. Rubens, C. Rigney, S. Willens, W. A. Simpson, Remote Authentication Dial In User Service (RADIUS), RFC 2865 (Jun. 2000). doi:10.17487/RFC2865.
[42] T. Dahm, A. Ota, dcmgash@cisco.com, D. Carrel, L. Grant, The TACACS+ Protocol, Internet-Draft draft-ietf-opsawg-tacacs-11, Internet Engineering Task Force, work in Progress (Sep. 2018).
[43] P. R. Calhoun, E. Guttman, J. Arkko, J. Loughney, Diameter Base Protocol, RFC 3588 (Sep. 2003). doi:10.17487/RFC3588.
[44] D. Simon, B. D. Aboba, P. Eronen, Extensible Authentication Protocol (EAP) Key Management Framework, RFC 5247 (Aug. 2008). doi:10.17487/RFC5247.
[45] A. Palekar, S. Josefsson, D. Simon, G. Zorn, Protected EAP Protocol (PEAP) Version 2, Internet-Draft draft-josefsson-pppext-eap-tls-eap-10, Internet Engineering Task Force, work in Progress (Oct. 2004).
[46] E. Rescorla, T. Dierks, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246 (Aug. 2008). doi:10.17487/RFC5246.
[47] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446 (Aug. 2018). doi:10.17487/RFC8446.
[48] E. Rescorla, N. Modadugu, Datagram Transport Layer Security Version 1.2, RFC 6347 (Jan. 2012). doi:10.17487/RFC6347.
[49] B. Campbell, C. Mortimore, M. Jones, Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants, RFC 7522 (May 2015). doi:10.17487/RFC7522.
[50] D. Hardt, The OAuth 2.0 Authorization Framework, RFC 6749 (Oct. 2012). doi:10.17487/RFC6749.
[51] F. Detienne, M. Kumar, M. Sullenberger, Flexible Dynamic Mesh VPN, Internet-Draft draft-detienne-dmvpn-01, Internet Engineering Task Force, work in Progress (Dec. 2013).
[52] D. Simon, D. B. D. A. Ph.D., P. Eronen, Extensible Authentication Protocol (EAP) Key Management Framework, RFC 5247 (Aug. 2008). doi:10.17487/RFC5247.
[53] H. Harney, A. Colegrove, U. Meth, G. Gross, GSAKMP: Group Secure Association Key Management Protocol, RFC 4535 (Jun. 2006). doi:10.17487/RFC4535.
[54] J. G. Myers, Simple Authentication and Security Layer (SASL), RFC 2222 (Oct. 1997). doi:10.17487/RFC2222.
[55] D. Piper, B. Swander, A GSS-API Authentication Method for IKE, Internet-Draft draft-ietf-ipsec-isakmp-gss-auth-07, Internet Engineering Task Force, work in Progress (Jul. 2001).
[56] J. Vilhuber, K. Kamada, S. Sakane, M. Thomas, Kerberized Internet Negotiation of Keys (KINK), RFC 4430 (Mar. 2006). doi:10.17487/RFC4430.
[57] M. Khalid, W. S. Wainner, A. Akhter, P. Quinn, VPN processing via service insertion architecture, US Patent 8,429,400 (Apr. 23 2013).
[58] D. McAlister, J. C. Orange, Protocol/API between a key server (KAP) and an enforcement point (PEP), US Patent App. 11/541,424 (Apr. 3 2008).

[59] M. L. Sullenberger, J. Vilhuber, Method and apparatus for establishing a dynamic multipoint encrypted virtual private network, US Patent 7,447,901 (Nov. 4 2008).
[60] Y. Nir, Q. Wu, An Internet Key Exchange Protocol Version 2 (IKEv2) Extension to Support EAP Re-authentication Protocol (ERP), RFC 6867 (Jan. 2013). doi:10.17487/RFC6867.
[61] F. Wei, C. Ma, Z. Zhang, Gateway-oriented password-authenticated key exchange protocol with stronger security, in: International Conference on Provable Security, Springer, 2011, pp. 366–379.
[62] K. Kamada, M. Ishiyama, S. Sakane, S. Zrelli, Problem Statement on the Cross-Realm Operation of Kerberos, RFC 5868 (May 2010). doi:10.17487/RFC5868.
[63] A. Bierman, M. Björklund, K. Watsen, RESTConf Protocol, RFC 8040 (Jan. 2017). doi:10.17487/RFC8040.
[64] H. Wang, X. Chen, Yang Data Model for Internet Protocol Security (IPsec), Internet-Draft draft-tran-ipsecme-yang-01, Internet Engineering Task Force, work in Progress (Mar. 2016).
[65] H. Soussi, M. Hussain, H. Afifi, D. Seret, IKEv1 and IKEv2: A quantitative analyses, in: Proceedings of World Academy of Science, Engineering and Technology, Vol. 6, 2005, pp. 194–197.
[66] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2, USENIX Association, 2005, pp. 273–286.
[67] M. Vajaranta, J. Kannisto, J. Harju, IPsec and IKE as Functions in SDN Controlled Network, in: Network and System Security, Springer International Publishing, 2017, pp. 521–530.
[68] Y. Rekhter, T. Li, A Border Gateway Protocol 4 (BGP-4), RFC 1654 (Jul. 1994). doi:10.17487/RFC1654.
[69] C. A. Sunshine, Survey of protocol definition and verification techniques, ACM SIGCOMM Computer Communication Review 8 (3) (1978) 35–41.
[70] E. Rescorla, Diffie-Hellman Key Agreement Method, RFC 2631 (Jun. 1999). doi:10.17487/RFC2631.
[71] R. Enns, M. Björklund, A. Bierman, J. Schönwälder, Network Configuration Protocol (NETConf), RFC 6241 (Jun. 2011). doi:10.17487/RFC6241.
[72] H. Feng, J. Llorca, A. M. Tulino, D. Raz, A. F. Molisch, Approximation algorithms for the NFV service distribution problem, in: INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, IEEE, 2017, pp. 1–9.