Tampere University

Ville Korhonen

FUTURE AFTER OPENVPN AND IPSEC

Computing and Electrical Engineering
Master of Science Thesis
August 2019

ABSTRACT

Ville Korhonen: Future after OpenVPN and IPsec
Master of Science Thesis
Tampere University
Master's Degree In Information Technology
August 2019

Virtual private networks (VPN) are used to connect private networks of organizations securely over public Internet. Virtual private network offers a secure and encrypted network connection even though the underlying network is public and insecure. In addition to connecting two remote sites with a virtual private network, it is possible to create a virtual private network between a remote worker and an organization site. Virtual private networks allow organizations to build secure connections relatively cheap and flexibly over the Internet.

The most widely used virtual private network protocols are IPsec (Internet Protocol Security) and SSL/TLS (Secure Socket Layer/Transport Layer Security) based protocols. IPsec is usually used for connecting two remote sites and SSL/TLS based technologies are used when a remote worker connects to the organization's resources. Both of these protocols have held a strong position for years even though especially IPsec have been criticized since the day it was born. The goal of this thesis was to investigate are there any alternatives available currently for IPsec and SSL/TLS based virtual private networks and are there protocols or architectures under development which could be alternatives in the future.

Based on the research, IPsec and SSL/TLS based protocols have superseded all older technologies and there aren't any real alternatives available to them. Many of the network devices manufacturers have IPsec based proprietary technologies which try to fix the issues of IPsec but they are not compatible with the technologies and products of other manufacturers. Out of the new protocols, Wireguard is promising and it might have a chance to challenge IPsec in the future. Software-defined networks (SDN) and software-defined perimeter (SDP) can also challenge traditional IPsec and SSL/TLS based virtual private networks when those solutions develop and become more common.

Keywords: VPN, IPsec, OpenVPN

TIIVISTELMÄ

Ville Korhonen: Tulevaisuus OpenVPN:n ja IPsecin jälkeen
Diplomityö
Tampereen yliopisto
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Elokuu 2019

Virtuaalisia erillisverkkoja (virtual private network, VPN) käytetään yhdistämään yritysten sisäverkkoja turvallisesti julkisen Internetin yli. Virtuaalinen erillisverkko tarjoaa turvallisen ja salatun verkkoyhteyden vaikka alla oleva verkko on julkinen ja turvaton. Kahden toimipisteen yhdistämisen lisäksi virtuaalinen erillisverkko voidaan muodostaa esimerkiksi työntekijän ja yrityksen toimipisteen välille. Virtuaalisten erillisverkkojen ansiosta organisaatiot voivat rakentaa yhteyksiä edullisesti ja joustavasti Internetin päälle.

Yleisimmät virtuaalisissa erillisverkoissa käytettävät protokollat ovat IPsec (Internet Protocol Security) ja SSL/TLS (Secure Socket Layer/Transport Layer Security)-pohjaiset protokollat. IPsec protokollaa käytetään yleensä yritysten toimipisteiden yhdistämiseen toisiinsa ja SSL/TLS-pohjaisia tekniikoita käytetään, kun työntekijä ottaa etäyhteyden organisaation resursseihin. Molempien asema on ollut hyvin vakaa jo useita vuosia, vaikka varsinkin IPseciä on kritisoitu sen syntymästä lähtien. Tässä diplomityössä pyrittiinkin selvittämään, onko IPsecille sekä SSL/TLS pohjaisille virtuaalisille erillisverkoille olemassa vaihtoehtoja ja onko kehitteillä protokollia tai arkkitehtuureja, jotka voisivat tulevaisuudessa olla vaihtoehtoja.

Selvityksen perusteella IPsec ja SSL/TLS-pohjaiset protokollat ovat syrjäyttäneet kaikki muut vanhemmat teknlogiat eikä niille ole ollut juurikaan vaihtoehtoja. Useilla verkkolaitevalmistajilla on omia IPsec pohjaisia ratkaisuja, jotka pyrkivät paikkaamaan IPsecin ongelmia, mutta niiden ongelma on yhteensopimattomuus muiden valmistajien tekniikoiden ja laitteiden kanssa. Uusista protokollista Wireguard on kuitenkin lupaava ja sillä voi olla mahdollisuus haastaa IPsec tulevaisuudessa. Lisäksi ohjelmisto-ohjatut verkot (software-defined networking, SDN) sekä ohjelmisto-ohjattu ulkoreuna (software-defined perimeter, SDP) voivat tekniikan kehittyessä ja yleistyessä haastaa perinteiset IPsec ja SSL/TLS-pohjaiset erillisverkkototeutukset.

Avainsanat: VPN, IPsec, OpenVPN

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

PREFACE

This thesis was done to the faculty of Computing and Electrical Engineering. I would like to thank Markku Vajaranta for the topic of this thesis and helping with the initial planning. I would also like to thank Marko Helenius and Evgeny Kucheryavy who reviewed this thesis.

Throughout the years spent in the university, I have spent several hours in TeLE's guild room. Thanks to the guild and people of it, my studies, work and social life have been in balance. I would also like to give special thanks Noora Hartikainen who provided the much needed peer pressure and peer support during the writing process.

In Tampere, 25th August 2019

Ville Korhonen

CONTENTS

1 Introduction 1
2 VPN tunnels 2
2.1 Techniques 2
2.2 IPsec 3
2.2.1 AH and ESP headers 4
2.2.2 ESP authentication and encryption algorithms 6
2.2.3 Security Associations 11
2.2.4 Security association and key management 11
2.3 SSL VPN 14
2.3.1 TLS protocol 15
2.3.2 SSL VPN Architecture 16
2.4 PPTP 17
2.5 L2TP 18
3 VPN Solutions 20
3.1 Commercial solutions 20
3.1.1 Cisco 20
3.1.2 Juniper 25
3.1.3 Pulse Secure 28
3.1.4 Checkpoint 29
3.1.5 Fortinet 29
3.1.6 Palo Alto Networks 30
3.2 Open source solutions 31
3.2.1 OpenVPN 31
3.2.2 IPsec 32
3.3 Comparison 33
4 Future 35
4.1 SoftEther 35
4.2 WireGuard 37
4.3 SDN and SD-WAN 40
4.4 SDP 42
5 Discussion 44
6 Conclusion 47
References 48

LIST OF FIGURES

2.1 Authentication Header 4
2.2 ESP in transport mode 5
2.3 ESP in tunnel mode 5
2.4 ESP format 6
2.5 CBC mode encryption 7
2.6 CBC mode decryption 8
2.7 CTR mode encryption 9
2.8 CTR mode decryption 9
2.9 AES-GCM encryption 10
2.10 IKE_SA_INT exchange with cookie challenge 13
2.11 IKE_AUTH exchange 13
2.12 TLS Handshake message flow 16
2.13 L2TP packet containing an IP datagram 18
2.14 L2TP datagram encrypted with IPsec 19

3.1 Hub-and-spoke configuration 22
3.2 Spoke-to-spoke configuration 23
3.3 GETVPN architecture 24
3.4 Hub-and-spokes topology with NHTB 26
3.5 OpenVPN multiplex model 32

4.1 SoftEther topology with cascading connection between office and data center . . . 36
4.2 Overview of handshake process in WireGuard when cookie is not used 38
4.3 Overview of handshake process in WireGuard when cookie is used 38
4.4 SDN architecture 41
4.5 SDP architecture 43

LIST OF TABLES

3.1 Routing and NHTB tables 26
3.2 Comparison of proprietary VPN solutions 34

LIST OF SYMBOLS AND ABBREVIATIONS

ADVPN	Auto Discovery VPN
AES	Advanced Encryption Standard
AH	Authentication Heade
ATM	Asynchronous Transfer Mode
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CTR	Counter mode
DES	Data Encryption Standard
DMVPN	Dynamic Multipoint VPN
DNS	Domain Name System
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ESP	Encapsulated Security Payload
FDB	Forwarding Database
GCM	Galois/Counter Mode
GDOI	Group Domain of Interpretation
GETVPN	Group Encrypted VPN
GRE	Generic Routing Encapsulation
HMAC	Hashed Message Authentication Code
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
L2TP	Layer 2 Tunneling Protocol
LDAP	Lightweight Directory Access Protocol
lsvpn	Large Scale VPN
MAC	Message Authentication Code

mGRE	multipoint Generic Routing Encapsulation
MPLS	Multiprotocol Label Switching
MPPE	Microsoft Point-to-Point Encryption
MSCHAP	Microsoft Challenge Handshake Authentication Protocol
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
NHC	Next Hops Client
NHRP	Next Hop Resolution Protocol
NHS	Next Hop Server
NIST	National Institute of Standards and Technology
OCVPN	Overlay Controller VPN
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
RC4	Rivest Cipher 4
RFC	Request For Comments
SA	Security Association
SD-WAN	Software Defined Wide Area Network
SDN	Software Defined Networking
SDP	Software Defined Perimeter
SNMP	Simple Network Management Protocol
SPD	Security Policy Database
SPI	Security Parameter Index
SSL	Secure Socket Layer
SSTP	Secure Socket Tunneling Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network

1 INTRODUCTION

Today's organizations are heavily utilizing networked resources and services. These services and resources are rarely hosted in the offices or all sites, but instead they can be hosted in own or partner's data centers or in multiple clouds. Most of the services can't be publicly available in the Internet so employees need to have a secure and limited way to access the resources from the offices and often even from home or other remote locations. To make these connections between sites and users possible and secure, virtual private networks (VPN) were invented.

IPsec and SSL/TLS based protocols are the most widely used protocols today [93]. They are relatively old protocols and the currently used VPN architectures which are based on them were developed in time before cloud based services started to appear. Therefore, they have some issues with scalability and security [26, 90].

This thesis was written to find out if there are any alternatives to the widely used IPsec and SSL/TLS protocols. Previous research about new alternatives was not found as the previous research has concentrated mainly on security and performance issues of IPsec and SSL/TLS VPNs and comparing these two protocols [4, 26, 50]. Upcoming protocols and new architecture models under development which could supersede the currently used protocols in the future were also explored in this thesis. The thesis was made as a literature review.

The thesis has six chapters. Chapter 2 covers the theory of IPsec and other commonly used protocols. In Chapter 3, we explore currently available and most widely used commercial and open source VPN solutions. Chapter 4 is dedicated covering the future technologies which are already available and under development. Discussion about the future technologies and current situation is carried out in the Chapter 5 and final conclusions are in Chapter 6.

2 VPN TUNNELS

Virtual private networks (VPN) are commonly used to interconnect private networks across the Internet as they allow users to send and receive data securely between remote private networks. Virtual private network connections can be built, for example, between two remote sites of an organization (site-to-site VPN) or between user and the organization (point-to-point VPN).

Site-to-site VPNs allow organizations to interconnect their offices over the Internet which is less expensive and more flexible than building dedicated network connections across long distances. When two sites are connected with VPN, computers and users in both sites are able to securely connect to resources in private networks of the other site.

Point-to-point VPNs are usually used by users, who need to access resources securely in their organization's private network from outside of their organization's network. Point-to-point VPNs allow employees to work from homes and other remote locations so point-to-point VPNs are widely used by all kinds of companies and organizations.

Usually, the purpose of a VPN connection is to ensure authenticity, integrity and confidentiality of the network traffic. When those three things are fulfilled, the connection is private and secure which means attackers can't eavesdrop, tamper or spoof traffic between VPN endpoints. How authenticity, integrity and confidentiality are achieved, depends on the used VPN technique.

2.1 Techniques

The two most common techniques for building virtual private networks are Internet Protocol Security (IPsec) and Secure Socket Layer Virtual Private Networks (SSL VPN). IPsec is usually used for building site-to-site connections and SSL VPNs are often used by remote users to create a point-to-point connection between their workstation and network of their organization.

There are also other techniques like Point-to-point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP) but they are not so widely used as IPsec and various SSL VPNs.

2.2 IPsec

Internet Protocol Security (IPsec) is a stack of protocols which is officially specified by the Internet Engineering Task force (IETF) in several Request for Comments (RFC) documents. It is used to secure Internet Protocol (IP) communications as it can encrypt, authenticate and checks the integrity of data packets of data streams. IPsec can be built between two routers, between a router and a host or between two hosts and it offers end-to-end protection for all applications as it operates on third level of OSI model.

Development of IPsec started originally in the early years of the 1990s, some time after the development of IPv6 was started. Development of IPv6 allowed IETF to define and integrate several security improvements to new versions of IP-protocol and so IPsec was defined as a part of IPv6 in RFC 1883. RFC 1883 only defines the requirements of IPsec as the protocols of the stack are defined in other RFCs [20]. Original RFC which defined the protocols of IPsec were RFC 1825-1829, which were published 1995 [8]. They have been superseded two times with newer versions and current, most essential ones, RFCs are 4301-4309 [49]. Originally, IPsec was developed for IPv6 but as it wasn't ready and widely used, IPsec was also developed for IPv4 and was widely deployed to IPv4 networks.

Like it was earlier said, IPsec is a protocol stack or framework which offers multiple services, algorithms and granularities. It offers services for data integrity, data origin authentication, confidentiality and protection against replay attacks [49]. That means user can select to use only those services which they need in their situation.

IPsec has multiple available algorithms as it was designed to be algorithm-independent because currently secure algorithms might not be secure forever and they may need to be changed to new ones. This has kept IPsec usable even though some of the algorithms it has used have been found out to be insecure [96].

Multiple granularities mean that users can create various configurations for tunnels depending on their needs. For example, users can have a tunnel which secures all traffic between two routers or all traffic between to hosts or have a separate encrypted tunnel for each TCP connection between two hosts [96].

IPsec can operate in two modes, transport and tunnel mode. The transport mode only encrypts the payload of IP packet as IPsec header is inserted after IP header. The transport mode can be used for connections between two hosts or between host and a gateway, so it is usually used for client-to-site IPsec VPN connections. If Encapsulating Security Payload (ESP) header is used, IPsec in transport mode can be used together with another tunneling protocol like Generic Routing Encapsulation (GRE) or L2TP. If it used with GRE or L2TP, another protocol encapsulates the IP data packet and IPsec is used to protect these encapsulated packets [107].

In tunnel mode IPsec protects the entire IP packet as it encapsulates the original packet in the payload of a new IP packet. As the entire IP packet is encapsulated and original source and destination addresses are hidden, tunnel mode hides the internal routing information.

The tunnel mode is usually used between two routers for interconnecting private networks securely over a public network [107]. Hosts in private networks don't need to be aware of IPsec as it is terminated in routers. That simplifies configuration as tunnel only has to be configured in routers. Compared to the transport mode and a connection without IPsec, tunnel mode increases packet size significantly as it adds an additional IP header.

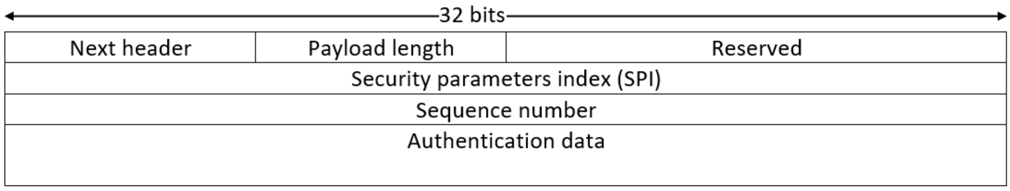
2.2.1 AH and ESP headers

Authentication Header (AH) and Encapsulating Security Payload (ESP) headers offer services for data integrity, data origin authentication, anti-replay attack and confidentiality.

- The IP Authentication Header ensures data integrity and data origin authentication, and it also offers optional anti-replay attack features [47].
- The Encapsulating Security Payload ensures data integrity and data origin authentication like AH and has anti-replay attack features, but also ensures confidentiality if user wants so [49].

AH and ESP may be used individually or together but using only ESP is the most common practice as most security requirements can be fulfilled with it. Both protocols support transport and tunnel modes.

The concept of Authentication Header is derived from SNMPv2 Security Protocol, which was defined in RFC 1446, and AH was originally defined in RFC 1826 [7, 30]. In IPv4 AH is inserted between IP header and Transport Control Protocol (TCP) header. In IPv6 AH is one of the extension headers. The format of an authentication header is displayed in the figure 2.1



Next header	Payload length	Reserved
Security parameters index (SPI)		
Sequence number		
Authentication data		

Figure 2.1. *Authentication Header*

- The Next Header of the authentication header is 8 bits wide and it contains the IP Protocol number of the original IP packet, because the protocol number of the original IP packet was replaced with 51 to indicate that authentication header follows it [47].
- The Payload length is 8 bits wide and it defines the length of authentication data [47].
- The Reserved field is reserved for future use, so currently it must be all zeros [47].

- The Security Parameters Index (SPI) is 32 bit field which contains pseudo-random value which is used to identify the security association of the datagram [47].
- The Sequence Number is a 32 bits wide field which contains a counter value. Value is increased by one for each sent packet. The counter is used for preventing replay attacks. This field is mandatory, even though replay attack protection is an optional service, so the sender must always include it in the packet. If the receiver doesn't want to use replay-attack protection, they can ignore the field. [47]
- The Integrity Check Value (ICV) field's length varies but it always contains a digital signature of the payload. The Security Association between the sender and the receiver defines the field's size and signature algorithm [47]. IPsec is based on symmetric cryptography, so sender and receiver have a shared key which is used for computing the signature.

The authentication header doesn't support encryption so it is useful only if data integrity is required but encryption is not needed. In addition to the payload, AH also checks the integrity of some other fields, including source address. Therefore, it ensures the origin of the data as attacker can't spoof the source address.

The ESP header is more commonly used than AH because it also offers encryption. Originally, AH offered only integrity checks like today and ESP only offered encryption, so they were used together. Later, ESP was developed and integrity checks were added to it which made AH obsolete [96].

Figures 2.2 and 2.3 show where the ESP header is located in transport and tunnel modes. Instead of doing integrity checks in ESP header, they are done with an additional field which comes after the payload, like shown in the figures 2.2 and 2.3.

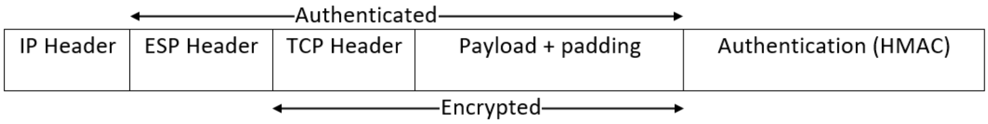


Figure 2.2. *ESP in transport mode*

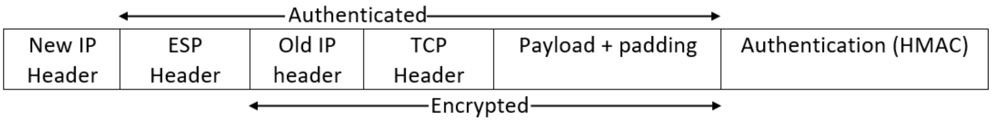


Figure 2.3. *ESP in tunnel mode*

The ESP header consists of two 32-bit words, which are Security Parameters Index and Sequence number. Both fields were also used in the AH header and they have an identical role as with AH: SPI contains a pseudo-random number which is used for identifying the security association of the datagram and sequence number field contains a counter that is increased for each sent packet to prevent replay attacks. ESP payload format is displayed in the figure 2.4.

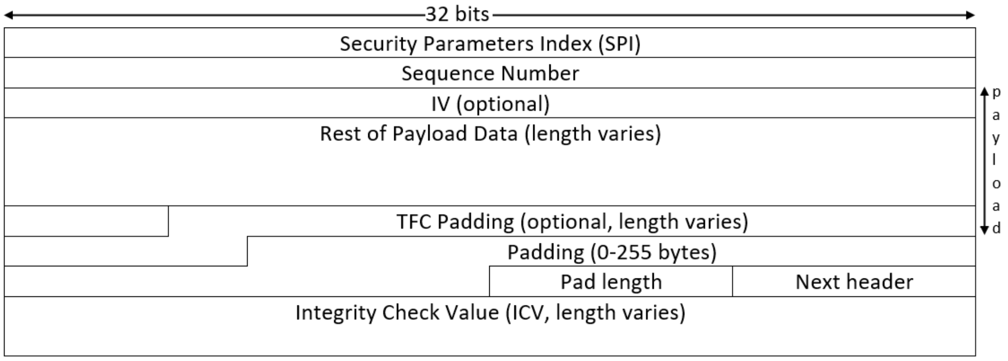

32 bits
Security Parameters Index (SPI)
Sequence Number
IV (optional)
Rest of Payload Data (length varies)
TFC Padding (optional, length varies)
Padding (0-255 bytes)
Pad length
Next header
Integrity Check Value (ICV, length varies)
payload

Figure 2.4. *ESP format*

After the ESP header, an ESP packet consists of the following fields:

- The Payload data field which starts with Initialization Vector (IV) for a cryptographic algorithm and continues with the contents of the original IP packet. If tunnel mode is used, there can also be Traffic Flow Confidentiality (TFC) padding at the end of the payload field. [48]
- The Padding field is 8 bits wide and it is used to extend the payload size to size that matches the cipher block size of the encryption. Length varies from 0 to 255 octets [48].
- The Pad Length is 8 bits wide and contains the length of padding in octets.
- The Next Header is 8 bits wide and it contains the IP Protocol number of the original IP packet [48].
- Integrity Check Value is a field with variable length which is computed from the ESP header, Payload and ESP trailer fields. This field is optional as it is only used if integrity service is enabled. Security Association between sender and receiver defines the field's size and signature algorithm. [48]

It is possible to use ESP in a authentication-only or encryption-only configuration but using encryption without authentication is not recommended as it is insecure [21, 72]. Using ESP in transport mode is less secure than using it in tunnel mode, as ESP in transport mode doesn't authenticate the original IP header, like the figure 2.2 shows.

2.2.2 ESP authentication and encryption algorithms

ESP can use various algorithms for encryption and authentication as protocol suite and ESP standard are independent of the cryptography algorithm [48, 49]. The supported and required algorithms of ESP and AH are listed in separate RFC, which have been updated a couple of times after it original release [1, 56, 59]. The current list of deprecated, required and recommended algorithms can be found in RFC 7321 [59].

Manufacturers tend to follow IPsec standard, so devices and applications have many different encryption and authentication algorithms available, so used algorithms are selected by user depending which algorithms are supported by tunnel endpoints and what are the requirements for security. As IP packets may arrive out of order and some packets may get lost, ESP standard says "each packet must carry any data required to allow the receiver to establish cryptographic synchronization for decryption" [48]. That data is usually carried in the payload field as Initialization Vector.

Currently, IPsec standard supports null, AES-CBC, AES-CTR and TripleDES-CBC algorithms for encryption, HMAC-SHA1-96, AES-GMAC with AES-128, AES-XCBC-MAC-96 and null algorithms for message authentication codes (MAC) and AES-GCM and AES-CCM for providing authentication and encryption in combined mode [59]. The old DES-CBC algorithm has been deprecated and current implementations of IPsec should not support it anymore as it has too short key and block size and therefore it is considered insecure [59]. TripleDES also has too small block size but it is still supported for backward compatibility and usage of it is discouraged [59]. Null algorithms are used, if either authentication or encryption is not required.

All currently recommended encryption and combined operation algorithms of IPsec use Advanced Encryption Standard (AES) cipher because it was selected by National Institute of Standards and Technology (NIST) as the standard for the encryption of electronic data. It replaced the old Data Encryption Standard (DES) which is currently considered as insecure.

AES-CBC mode uses Advanced Encryption standard Cipher in Cipher Block Chaining mode with an Initialization Vector. IV must be as long as the block size. AES-CBC mode supports 128 bit, 192 bit and 256 bit keys, but only 128 bit keys are mandatory for the IPsec standard [29]. Figures 2.5 and 2.6 shows how CBC mode encrypts and decrypts data.

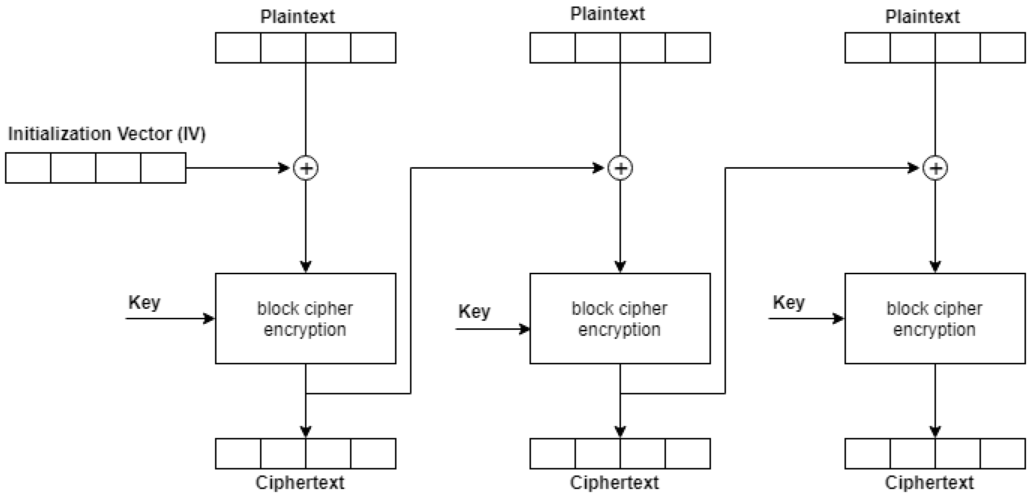


Figure 2.5. *CBC mode encryption*

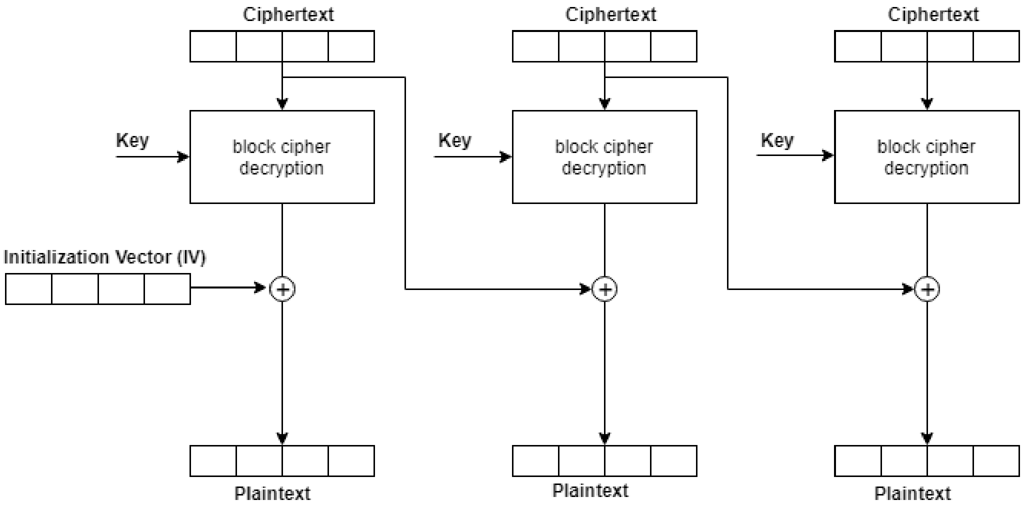


Figure 2.6. *CBC mode decryption*

The CBC mode is the oldest and most commonly used block cipher operation mode. In RFC 7321, it is the only encryption algorithm which is mandatory for IPsec implementation [59]. The main drawbacks of CBC-mode are that encryption cannot be parallelized and that messages must be padded to a multiple of cipher block size. Lack of parallelizing hurts the performance which can be an issue with IPsec tunnels if high throughput is required from a tunnel.

AES-CTR mode uses AES cipher in Counter mode, which turns a block cipher into a stream cipher. CTR mode requires a counter value which consists of a nonce, an Initialization Vector and a counter block. Nonce is a single use value which is assigned at the beginning of security association, Initialization Vector is a unique value generated by sender for each used key and block counter is a value which is increased after each packet to generate next parts of the key stream. [37]

Figures 2.7 and 2.8 illustrate how CTR mode encryption works. Decryption is the same as encryption, but backwards, so separate decryption implementation is not required.

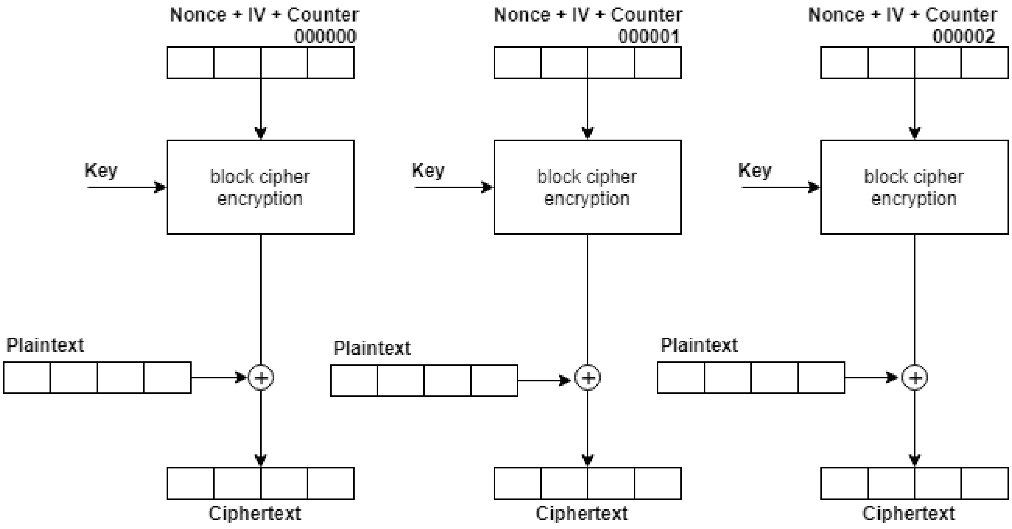


Figure 2.7. *CTR mode encryption*

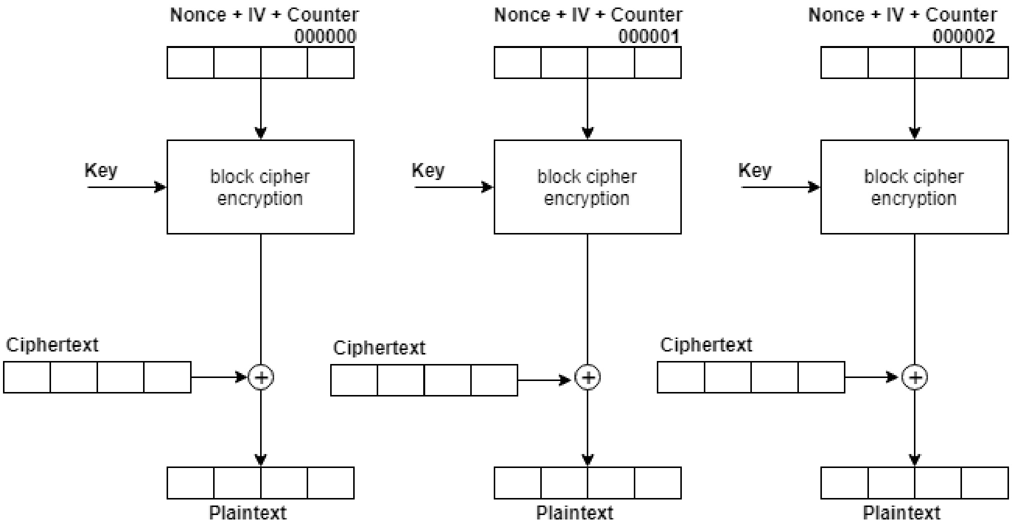


Figure 2.8. *CTR mode decryption*

Compared to the CBC mode, the CTR mode can be faster as the encryption process can be parallelized and the sender can precompute the key stream which makes it suitable for high-speed networking [55]. Another advantage of CTR mode is that it doesn't require padding. The disadvantage of CTR mode is that it is easily misused: if Initialization Vector is used for more than one packet with the same key, the confidentiality can no longer be guaranteed [37]. The CTR mode must not be used without authentication as it is easy construct a forged ciphertext from a valid ciphertext [37].

The CTR mode would perform great with IPsec tunnels if it wouldn't require a separate authentication algorithm. CTR mode itself is parallelizable and therefore fast, but available authentication algorithms are not fast enough for high data rates [100].

The above mentioned CBC and CTR modes are encryption only algorithms and therefore they should always be used together with authentication algorithms which IPsec offers [59]. If either AES in Counter with CBC-MAC (AES-CCM) mode or AES in Galois/Counter

(AES-GCM) mode is used in combined mode, a separate authentication algorithm is not required as these algorithms can also authenticate the packets [38, 100].

AES-CCM mode uses AES cipher in Counter with Cipher Block Chaining Message Authentication code mode [38]. AES-CCM is a stream cipher like AES-CTR, but before encryption it first computes authentication value with CBC-MAC and then encrypts the payload with AES in Counter mode [108]. Performance is about the same as with CBC mode [108].

AES-GCM uses AES in Galois/Counter mode. Like CCM mode, it is based on CTR mode, but instead of using CBC-MAC, it uses a binary Galois field to provide authentication [58]. The advantage of AES-GCM is that it combines the fast CTR mode with an authentication mechanism which can be parallelized [100]. Therefore, AES-GCM is the fastest mode for IPsec encryption and authentication.

Figure 2.9 illustrates how GCM-mode works. E_K denotes the used block cipher using the key K, $mult_H$ denotes multiplication by the hash key H and incr denotes counter increment function [58].

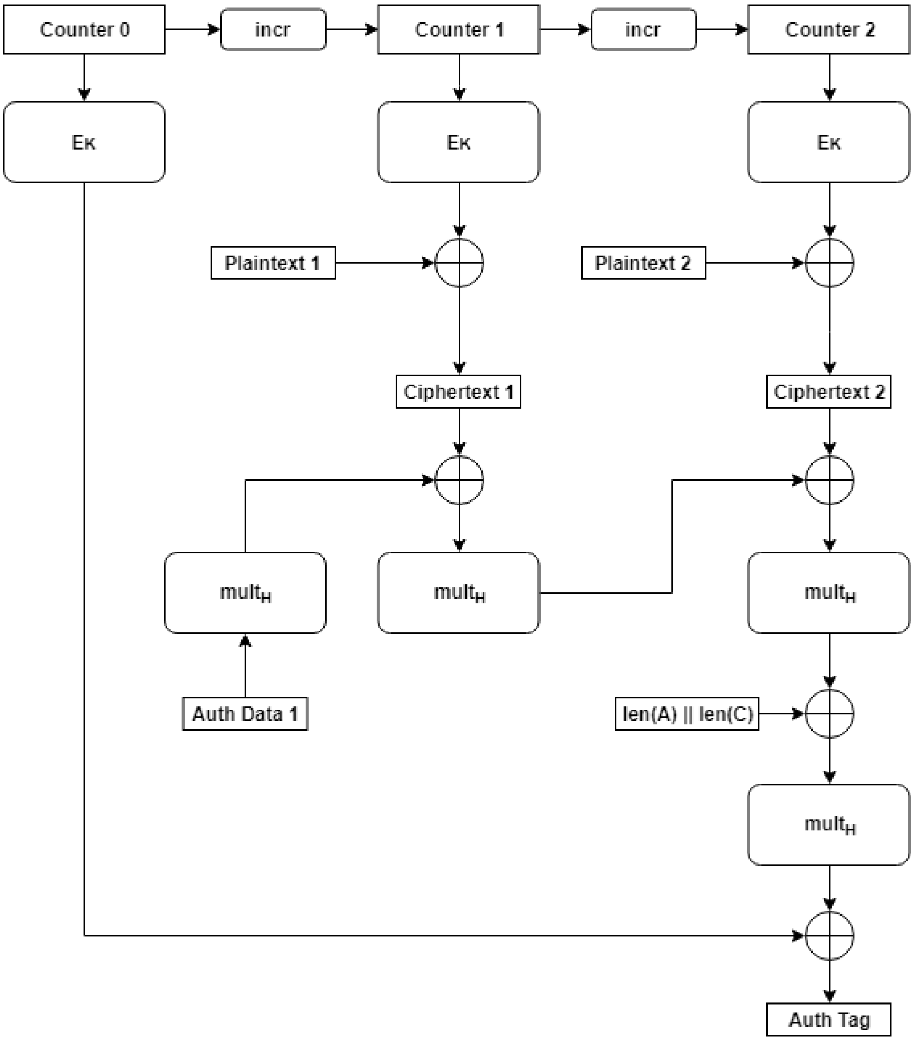


Figure 2.9. *AES-GCM encryption*

Because the GCM mode is based on the CTR mode, it shares the same security issues, so the combination of a key and an IV must not be used more than once. Therefore all AES-GCM implementations for IPsec must use automated key management [100].

2.2.3 Security Associations

Security Association (SA) is a logical simplex connection between IPsec endpoints which has to be established before endpoints can exchange any data over IPsec. If both endpoints want to send and receive data, they need two Security Associations as connections are simplex. [49]

Security Associations contain the security parameters which are required for sending and receiving packets to the other endpoint. SA contains the following data:

- Security Parameter Index (SPI) which is used to identify the SA
- The Source and destination addresses of the SA
- Used encryption algorithm and mode
- Encryption key
- Used authentication algorithm
- Authentication key
- Additional attributes like SA lifetime

Information about Security Associations is stored stored in Security Association Database (SAD) [49]. When IPsec sends data to another endpoint, it first consults Security Policy Database (SPD) and according to the first matching SPD rule, the packet is either discarded, sent without IPsec protection or sent to SA found in the SPD rule [49]. When packets are sent to SA, they are then authenticated and encrypted with parameters defined by SA [49, 51]. If SA doesn't exist, IKE process to create SA is started.

Incoming packets are matched to correct SA in SAD using a combination of SPI, source address and destination. When matching SA is found, packets are authenticated and decrypted with parameters found in SA [49].

2.2.4 Security association and key management

Security Associations can be configured manually but that is not recommended as it makes management of multiple IPsec tunnels difficult. The most serious issue with a manual configuration is that secret keys of Security Associations should be used only for a limited time and if the configuration has been done manually, the secret key has to be changed manually. To solve issues with manual configuration of Security Associations, IPsec supports Internet Key Exchange (IKE) protocol which allows dynamic configuration of Security Associations [46].

The latest and currently recommended version of IKE is the version 2 (IKEv2). First version of IKE (IKEv1) was described in RFC 2409 in 1998 and it was replaced with IKEv2 which was described in RFC 4306. IKEv2 has been updated a couple of times and the current version is defined in RFC 7296 [46]. IKEv1 is still widely supported by networking devices and software but using it is not recommended as it has several flaws [73].

All IKE communication always consists of requests and responses to them and these two messages form an exchange. If requester doesn't get response within specified timeout, it should resend the request or discard the connection completely. [46]

The IKE process starts with IKE_SA_INT exchange. During the first exchange, participants negotiate cryptographic algorithms, exchange nonces, execute Diffie-Hellman exchange and open IKE Security Association (IKE SA) which is a secure and bi-directional communication channel between endpoints. That channel is used in IKE phase 2. [46]

IKE_SA_INT exchange has either 2 or 4 messages. In the basic operation mode, only two messages are required but that leaves protocol open for spoofing attacks [41]. To protect against spoofing, IKEv2 can use additional 2 messages for cookies during the IKE_SA_INT exchange. Figure 2.10 displays the course of the exchange with cookie challenge.

Abbreviations used in the figures 2.10 and 2.11 are described below:

- AUTH denotes authentication date of the sent message
- CERT denotes optional certificate
- CERTREQ optional certificate request
- HDR denotes IKE header
- IDi denotes identification of initiator
- IDr denotes identification of responder
- KEi denotes Diffie-Hellman key exchange value of initiator
- KEr denotes Diffie-Hellman key exchange value of responder
- Ni,Nr denotes nonce
- SAi1 denotes payload containing cryptographic algorithms supported by initiator
- SAr1 denotes payload containing the selected cryptography algorithms
- SAi2 denotes payload containing data of offered SA
- SAr2 denotes payload containing accepted offer
- TSi denotes Traffic Selector of initiator
- TSr denotes Traffic Selector of responder

Cookies are used if a certain threshold of incomplete sessions is reached and the responder decides to generate a cookie and send it to the initiator. The initiator must attach the cookie to the original message to prove the original message wasn't spoofed. When the responder receives the request with a valid cookie, it replies to the initiator and completes the exchange.

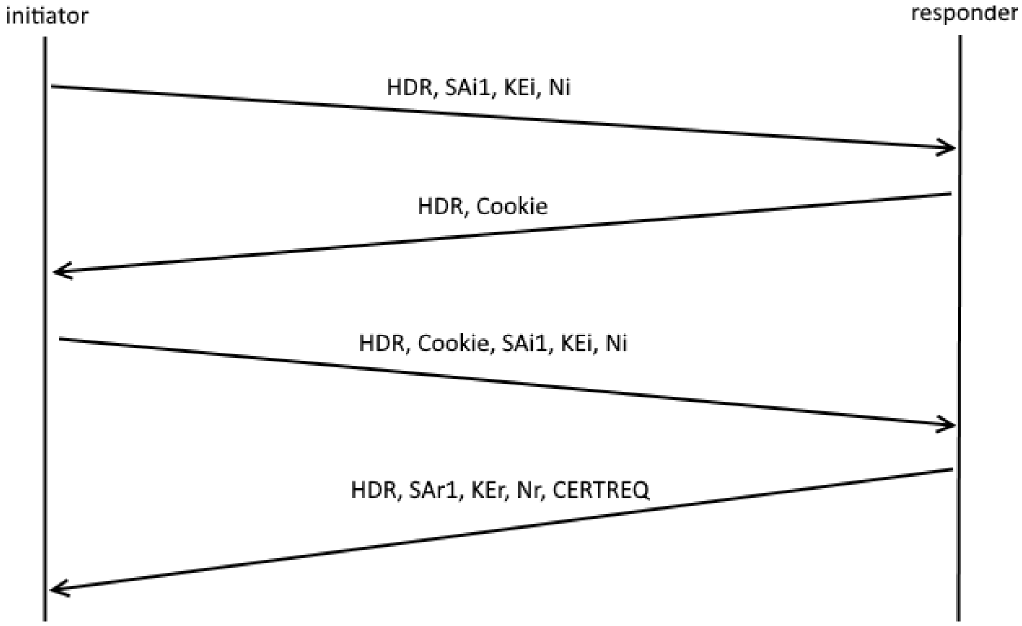


Figure 2.10. *IKE_SA_INT exchange with cookie challenge*

The process continues with IKE_AUTH exchange. All messages of the exchange are sent over the secure IKE SA which was created after IKE_SA_INT exchange. During the IKE_SA exchange both sides of the connection reveal their identities to authenticate themselves, negotiate the encryption and authentication algorithms and establish encryption and authentication keys [46]. Exchange is illustrated in the figure 2.11.

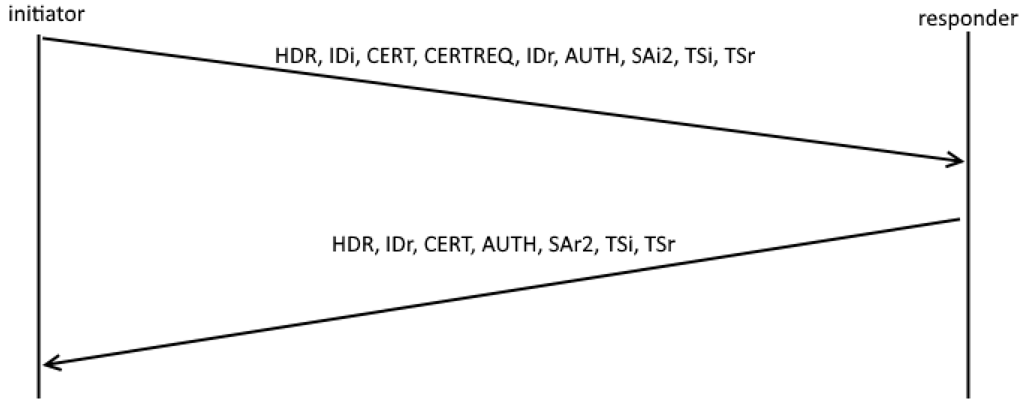

initiator
responder
HDR, IDi, CERT, CERTREQ, IDr, AUTH, SAi2, TSi, TSr
HDR, IDr, CERT, AUTH, SAr2, TSi, TSr

Figure 2.11. *IKE_AUTH exchange*

TSi and TSr payloads are used transmit Traffic Selector (TS) payloads from endpoint to other. TS payloads allow endpoints to communicate some information from their SPD to each other. TS payloads specify which packets are forwarded to which SA. In some scenarios, this can be used to check consistency between SPDs and in other scenarios these payloads can be used to dynamically update SPDs. [46]

TSi payload specifies the source addresses and TSr payload specifies the destination addresses of traffic which is forwarded to the SA from the initiator. If the responder accepts the proposal of the initiator, it sends identical TSi and TSr payloads back to the initiator [46]. If responder doesn't have identical Traffic Selectors, it can narrow them to a subset of the proposal:

- If the responder doesn't accept any part of the proposed Traffic Selectors, it responds with TS_UNACCEPTABLE notify message [46].
- If the responder accepts the entire TSi and TSr, narrowing the proposal is not required and the responder can reply with identical TSi and TSr values [46].
- If the responder accepts the first selector of TSi and TSr, then responder narrows the Traffic Selectors to a subset that includes the initiator's proposal. For example, if proposed TSi contained subnet 192.168.1.0/24 and TSr contained subnet 192.168.2.0/24, responder can respond identical TSi and with TSr set to 0.0.0.0/0. [46]
- If the responder do doesn't accept the first selector of TSi and TSr, it narrows them to an acceptable subset [46]

After IKE_AUTH exchange is completed successfully, a pair of simplex IPsec SAs is created. These Security Associations are also called child SAs. [46]

In addition to IKE_INT and IKE_AUTH exchanges, IKE has CREATE_CHILD_SA and INFORMATIONAL exchanges. CREATE_CHILD_SA is used when new child SA is needed or one of the existing child SAs needs to be re-keyed. Re-keying is used because encryption keys should be used only for a limited time. Re-keying creates a new child SA with new keys, moves traffic to new child SA and finally deletes the old child SA with INFORMATIONAL exchange. Usage of re-keying process instead of deleting and creating a new SA prevents the breaks of the IPsec connection. [46]

INFORMATIONAL exchanges can have three kinds of payloads: Delete payload is used for deleting child SAs, Notify payload can carry error and status information and Configuration payload is used for exchanging configuration between peers [46].

2.3 SSL VPN

There isn't any single standard defining SSL VPNs. Instead of a single standard, several software and networking equipment companies have their own products which all create SSL VPN tunnels. Lack of single standard means that products of different vendors are not usually compatible with each other. Even though there isn't any single standard for SSL VPNs, they all use Transport Layer Security (TLS) protocol. TLS superseded the original Secure Socket Layer (SSL) protocol [22], but SSL acronym is still used with SSL VPNs.

2.3.1 TLS protocol

TLS protocol is widely used for securing TCP traffic. The predecessor, SSL, was originally designed by Netscape engineers and it was first used to protect traffic between web servers and web browsers [51]. Later the usage of it has spread to other purposes, like SSL VPNs, because SSL/TLS protocol can be used to protect any kind of traffic that uses TCP [51]. SSL/TLS offers encryption using symmetric cryptography, authentication using public-key cryptography and integrity for the traffic using message authentication code [22] so it fulfills the basic requirements of a VPN.

The TLS protocol consists of four steps: handshake and the cipher suite negotiation, authentication of parties, key-related information exchange and application data exchange [99]. First three of those steps are handled by TLS Handshake Protocol and application data exchange is handled by TLS Record Protocol [99].

A TLS handshake starts when a client sends Client Hello message to the server. A Client Hello message contains protocol version, a client random, a session ID, supported cipher suites and supported compression methods. The server tries to find an acceptable set of algorithms from the Client Hello message and if it success, it will reply with Server Hello message. Server hello message contains protocol version, a server random, a session ID, the selected cipher suite and the selected compression method. [22].

After the server has sent a hello message, it sends the server certificate in Certificate message. If the server doesn't have a certificate, it sends a Server Key Exchange message to the client. Additionally, the server can request a certificate from the client. After these messages, the server sends a Server Hello Done message which indicates first part of the handshake is complete and waits reply from the client. [22].

If the server requested a certificate from the client, the client sends a certificate to the server. Immediately after the client has sent a certificate, it sends a Client Key Exchange message which contains an encrypted premaster secret or Diffie-Hellman parameters for agreeing upon the same premaster secret. If the client certificate has signing capability, it sends a signed Certificate Verify message to the server. [22].

When the both client and server have the premaster secret, they use both use the same key derivation function to compute master secret (MS) from the premaster secret and random values which were part of the hello messages. Master secret is divided into two encryption and two MAC keys: one encryption and MAC key for the client and one encryption and MAC key for the server [22].

After the client has computed the encryption and MAC keys and has sent the certificate and the subsequent messages, it continues with a Change Cipher Spec message. When the message has been sent, it starts to use the agreed cipher suite and computed keys and sends a Finished message encrypted with the selected algorithm and key. Server replies to these messages with its own Change Cipher Spec message and finishes handshake process by sending Finished message encrypted with the selected algorithm and key. [22].

When the handshake has been completed successfully, the client and the server can start sending application data via secure message channel. The message flow of handshake process is illustrated in the figure 2.12.

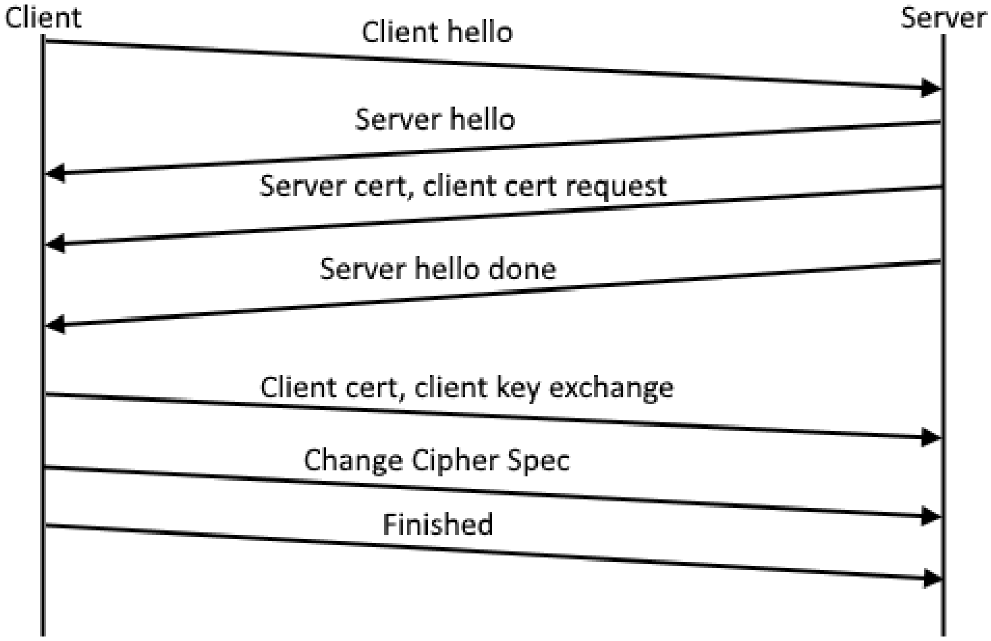


Figure 2.12. *TLS Handshake message flow*

Securing the application data is the responsibility of the record protocol [99]. Even though TLS runs over TCP and TCP is a byte stream protocol, TLS doesn't encrypt and send data as a data stream because including MAC in a stream wouldn't be possible [51]. Therefore, the TLS record protocol, which is responsible for securing the application data, splits data into manageable blocks, compresses the data if required, applies MAC to the data block, encrypts it and then passes it to TCP for transmit [22]. In the receiving end, record protocol decrypts data block, verifies the integrity and origin of the data, decompresses and reassembles it and finally passes it to the application which is using TLS [22].

2.3.2 SSL VPN Architecture

SSL VPNs usually support two different operating modes: portal and tunnel mode [90, 91, 92]. In portal mode, users connect to a web portal with a web browser and after login, they can access services through the portal page. Using the portal mode is easy for users as they only need a web browser which supports TLS and credentials, but the portal mode has its limitations: users can only access services and resources via the VPN portal with browser and all other traffic is routed towards public Internet [90, 91].

The tunnel mode is the other operating mode for SSL VPNs and it usually requires a client software to the user's workstation. A client software or a web browser connects to the VPN endpoint, authenticates the user and creates a TLS tunnel between workstation and the VPN endpoint. Users can be authenticated with combination of X.509 certificates, username, password and one-time-password [6, 18, 31]. Depending on the configuration all network traffic can be routed through the VPN tunnel or just the traffic which destination is in the intranet of the organization. The advantage of the tunnel mode is that it allows users to use all kinds of applications to access resources and services in the intranet of the organization. If all traffic is routed to the tunnel, organization can monitor and protect their VPN users with organization firewalls and other security appliances.

2.4 PPTP

Point-to-point Tunneling Protocol (PPTP) is a protocol for tunneling PPP packets over IP networks developed in the 1990s by several companies, including Microsoft and 3Com. It was designed to be used as a VPN protocol and it was widely used before it was found to be insecure. Protocol is defined in RFC 2637, but it has not been ratified by IETF as a standard [35].

PPTP uses TCP control channel and GRE tunnel to encapsulate the PPP packets. Traffic is first encapsulated to PPP packets, then those packets are encapsulated with GRE and those are sent over IP network to the VPN endpoint which extracts the original payload from the encapsulated packets. Every GRE tunnel has a separate TCP control channel, which is used for the establishment, management and release of the connection. [35]

PPTP specification doesn't define any authentication or encryption methods as it relies on Point-to-Point Protocol (PPP) on these things. PPP neither defines any algorithms for authentication or encryption, but it offers a framework for negotiating them [60, 76, 81]. Most of the algorithms which are used by PPTP have been developed by Microsoft as they were part of the PPTP development consortium and most of the commercial PPTP solutions have been published and distributed by Microsoft. All Microsoft Windows versions have supported PPTP since Windows 95.

It was already found in 1998 that Microsoft's authentication protocol Microsoft Challenge Handshake Authentication Protocol (MSCHAP) and RC4 based encryption protocol (MPPE) are weak and easy to crack [78].

After issues were found in MSCHAP and MPPE, Microsoft released MSCHAP v2 and updated MPPE protocol to use different keys to each direction [79]. Changes fixed the most critical security flaws found found earlier from the protocols, but like Schneider and Mudge mentioned in their analysis, the fundamental flaw of the authentication and encryption algorithms is that they are only as strong as user's password [79].

In 2012, it was demonstrated that brute-force cracking of MSCHAP v2 key is as simple as cracking a DES key and it will take only 23 hours to crack it with an online service [57].

Computing power has increased since that so cracking MSCHAP v2 is even faster today and therefore PPTP should be used no more as all traffic can be decrypted relatively easy. However, it is still probably used in some older environments and some public VPN providers still support it.

Above mentioned security flaws are not flaws of the PPP and PPTP protocols itself as they are only issues of Microsoft protocols. Microsoft protocols are practically only authentication and encryption protocols for PPTP which are widely supported by operating systems and networking devices, so there aren't any alternatives to them and therefore there are no options to make PPTP secure.

2.5 L2TP

Layer 2 Tunneling Protocol (L2TP) is a protocol for tunneling and transmitting datagrams of L2 protocols over various networks like IP and ATM [66]. Basic concept is quite similar to PPTP and it uses PPP like PPTP [66]. The latest version L2TPv3 also supports other L2 protocols than PPP like Ethernet and Frame Relay [53]. Like PPTP, L2TP doesn't provide any confidentiality or encryption by itself. Structure of L2TP packet containing an IP datagram is shown in picture 2.13.

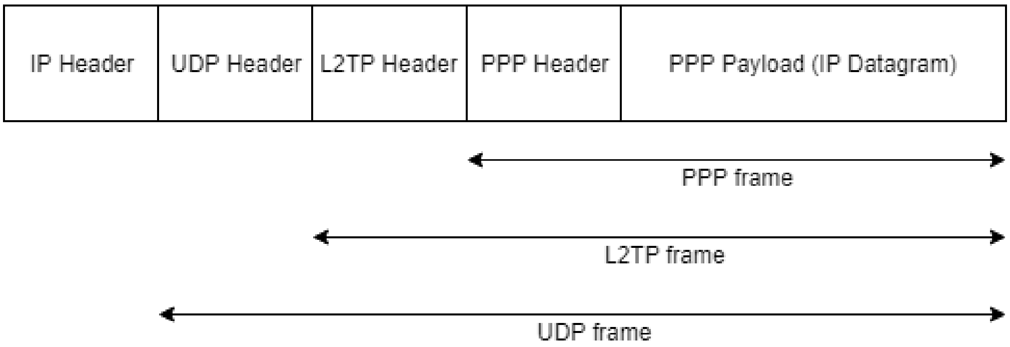


Figure 2.13. *L2TP packet containing an IP datagram*

As L2TP uses PPP, it inherits the PPP authentication, encryption and compression protocols, but they don't fulfill the security requirements: PPP authentication doesn't provide per packet authentication, integrity or replay protection and PPP encryption doesn't offer authentication, integrity checks or replay protection. Therefore, RFC 3193 proposes that IPSec suite should be used for protecting L2TP traffic in IP networks. [71]

The combination of L2TP and IPsec is often called L2TP/IPsec. It works as plain IPsec but instead of encapsulating IP Datagrams, it encapsulates L2TP datagrams which contains the payload. Structure of encapsulated L2TP datagram is shown in picture 2.14.

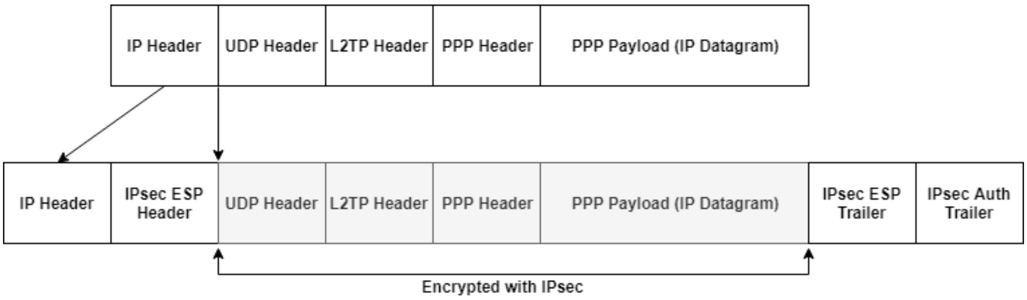


Figure 2.14. *L2TP datagram encrypted with IPsec*

Compared to plain IPsec, L2TP/IPsec has more overhead as L2TP encapsulation adds additional headers to the encapsulated datagram so performance can be worse than with plain IPsec. The main reason to use L2TP/IPsec is that it allows transmitting L2 traffic over IPsec tunnel which plain IPsec doesn't allow.

3 VPN SOLUTIONS

3.1 Commercial solutions

Several networking equipment and software vendors offer products which allow organizations to build site-to-site and client VPNs for their users. All of the client VPN products support SSL/TLS encryption and some of them also support IPsec. For site-to-site VPNs, IPsec is the most widely supported and used technology.

Even though all commercial client VPN solutions support SSL/TLS and some cases IPsec, they are incompatible with the products of other vendors as everyone has their own software for their VPN. That means organizations are locked to the server and client software of a single vendor. Therefore, organizations must consider the pros and cons of both server and client software before choosing a solution.

Basic site-to-site IPsec solutions should be compatible between all vendors as IPsec is a standard but some vendors might have features which others don't support and some configuration sets can be incompatible with some devices or software versions. Some vendors also have their own proprietary site-to-site VPN solutions which are only supported by their devices.

3.1.1 Cisco

Cisco Systems, Inc. is the largest networking company in the world. They develop, manufacture and sell all kinds of networking hardware, software and services to enterprises. Cisco currently has six different VPN solutions in their portfolio:

- Dynamic Multipoint VPN (DMVPN) is dual-stacked (IPv4/IPv6) solution which uses multipoint Generic Routing Encapsulation (mGRE) and IKEv1 or IKEv2 for building scalable IPsec networks. DMVPN has a centralized architecture so all tunnels have the same endpoint in the central hub and according to Cisco, connecting new sites to the central location should be simple as the hub doesn't require any changes to the configuration.
- FlexVPN is an IPsec based solution which aims to simplify the deployment of VPNs for remote access, site-to-site and other scenarios. It can be considered as a developed version of DMVPN as it supports the same features and configuration is

simplified compared to the DMVPN.

- Group Encrypted Transport VPN (GETVPN) is a VPN solution which allow encrypting MPLS and IP WANs without losing any-to-any connectivity and simplifies encryption management through the use of group keying instead of point-to-point key pairs.
- SSLVPN is a VPN solution for remote access which uses TLS or Datagram Transport Layer Security (DTLS) for encrypting the data.
- Easy VPN is Cisco's proprietary solution based on IPsec which aims to simplify VPN deployments for remote sites and mobile workers.
- Standard IPsec and Static IPsec are terms of Cisco for standard site-to-site IPsec connection between two locations.

DMVPN is Cisco's proprietary solution to building GRE over IPsec tunnels in dynamic and scalable manner. It relies on two relatively old technologies: Next Hop Resolution Protocol (NHRP) and mGRE. [14]

Multipoint GRE simplifies the tunnel configuration significantly as it allows single GRE interface to support multiple tunnels. As only one interface is required to the central hub, the amount of configuration in smaller than when using one GRE interface per tunnel [14].

NHRP allows systems connected to Non-Broadcast Multi-Access (NBMA) network to learn the physical NBMA address of the other systems that are part of the network, allowing them to directly communicate with each other. NHRP utilizes Next Hop Server (NHS) which is the central hub for communication. Other nodes of the network are spokes for it and they act as Next Hop Clients (NHC). The NHS maintains a NHRP database of the public interface addresses of each spoke and when a spoke wants to connect to other spoke, it queries NHS for the address of the other spoke. [13]

DMVPN provides full or partial meshed connectivity between the hub and spokes. Only the hub requires static IP addresses and adding new spokes doesn't require any changes to the hub as multipoint GRE can handle new tunnels from new spokes automatically. Hub-to-spoke tunnels are static but DMVPN can also dynamically create direct tunnels from spoke-to-spoke. [14]

Hub-and-spoke configuration, which is displayed in the figure 3.1, is the simplest setup for DMVPN. All traffic between spokes goes through the hub and every spoke requires their own tunnel. Tunnels between spokes and the hub are static but compared to using normal GRE over IPsec tunnels, configuration is simpler as adding new spokes doesn't require any additional configuration to hub. [14]

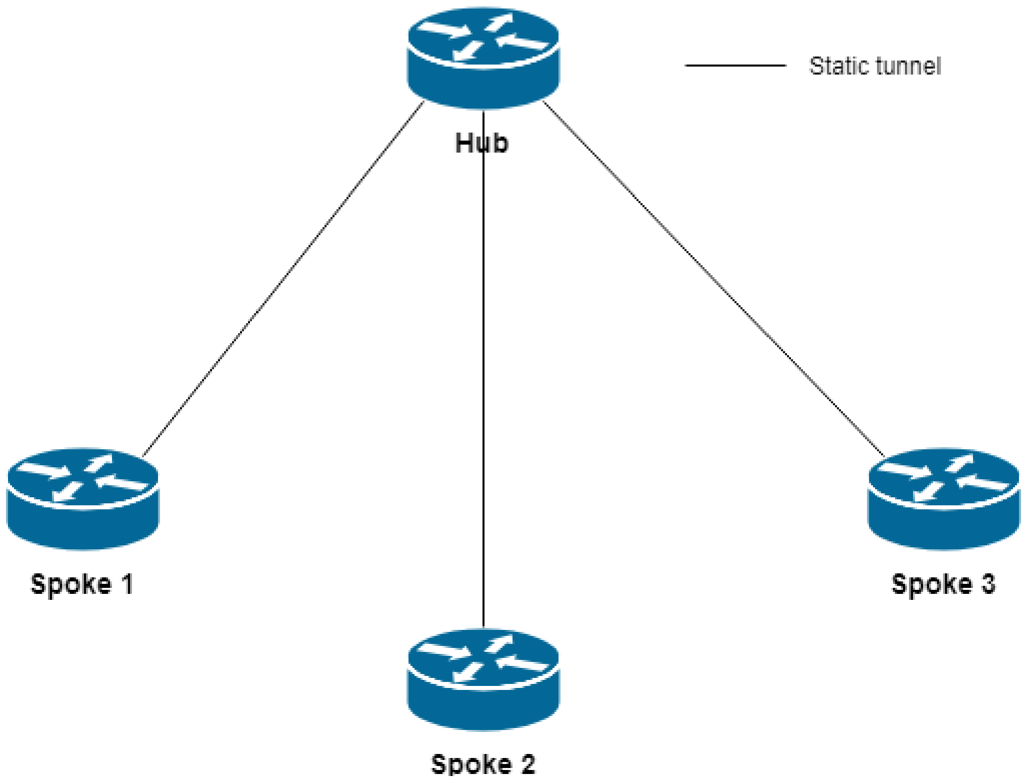


Figure 3.1. *Hub-and-spoke configuration*

In a spoke-to-spoke configuration, which is shown in the figure 3.2, tunnels between spokes and hub are static tunnels like in the hub-to-spoke configuration. Tunnels between spokes are created dynamically when spokes need to communicate with each other. When spoke needs to connect to another spoke, it queries the NHS on the hub and requests the NHRP mapping from the NHS. Source spoke gets the NHRP mapping, which tells the public interface address of the destination spoke, from the NHS and with the help of that mapping it can create a direct tunnel to the destination spoke dynamically. [14]

Using dynamic tunnels between spokes reduces latency and load on the hub as unicast traffic between the spokes don't need to go through the hub. However, multicast and control traffic is still sent through the hub. [14]

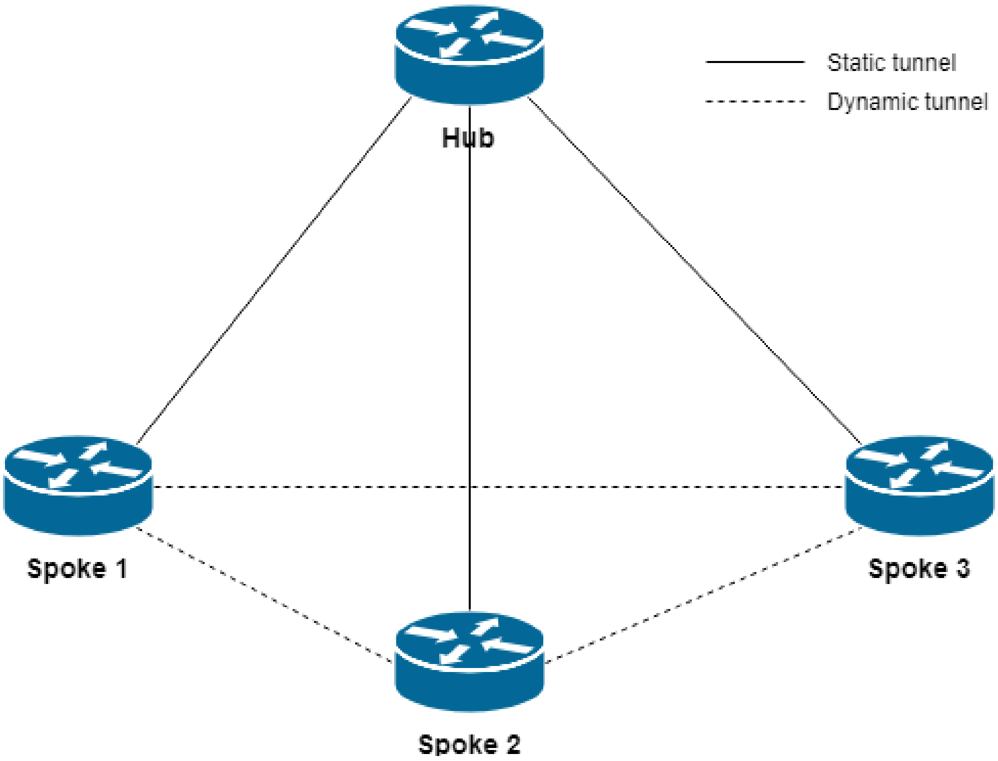


Figure 3.2. *Spoke-to-spoke configuration*

Cisco also offers solution called FlexVPN which is in practice a more developed version of DMVPN. It supports site-to-site, hub-to-spoke and spoke-to-spoke tunnels like DMVPN and in addition to those, FlexVPN also supports remote-access setups for teleworkers. Both are based on the same basic technologies: IPsec, GRE and NHRP. There are still some major differences between the two solutions.

FlexVPN uses multiple static and dynamic GRE interfaces instead of one static mGRE interface which DMVPN uses. Usage of multiple GRE interfaces instead of one mGRE interface gives better control over each hub-to-spoke connection and therefore allows more customized configurations for each connection and more flexible QoS options. Interfaces are static for hub-to-spoke connections and dynamic for spoke-to-spoke connections. Dynamic interfaces are cloned from a virtual-template interface.

In FlexVPN, spokes don't register to the hub like in DMVPN as NHRP is only used to establish spoke-to-spoke connections. As NHRP is not used for hub-to-spoke registration, routing information needs to be advertised via other mechanisms. FlexVPN supports IKEv2, so routing information can be inserted into IKEv2 SAs for route advertising or dynamic routing protocols like BGP can be used to advertise routes.

Another alternative to the DMVPN and FlexVPN is Cisco's GETVPN. It is completely different compared to all other VPN solutions in Cisco's portfolio as it tunnel-less VPN solution which is based on Group Domain of Interpretation (GDOI) and IPsec. Compared to DMVPN and traditional IPsec tunnels, GET VPN is more scalable as point-to-point tunnels are not required, it supports native routing and multicast works without any issues.

The key technology behind GETVPN is GDOI. GDOI is a protocol defined in RFC 3547 which handles management of cryptographic keys and policies for the devices participating in GET VPN infrastructure.

All VPN endpoints, which are called Group Members (GM) in GET VPN, are part of a trusted group which shares a group security association (group SA). As SA is shared, all of the group members share the same encryption keys and parameters and therefore all GMs can decrypt the messages of other GMs of the group. Shared SA allows GMs to communicate with each other without point-to-point tunnels and this makes GET VPN scalable.

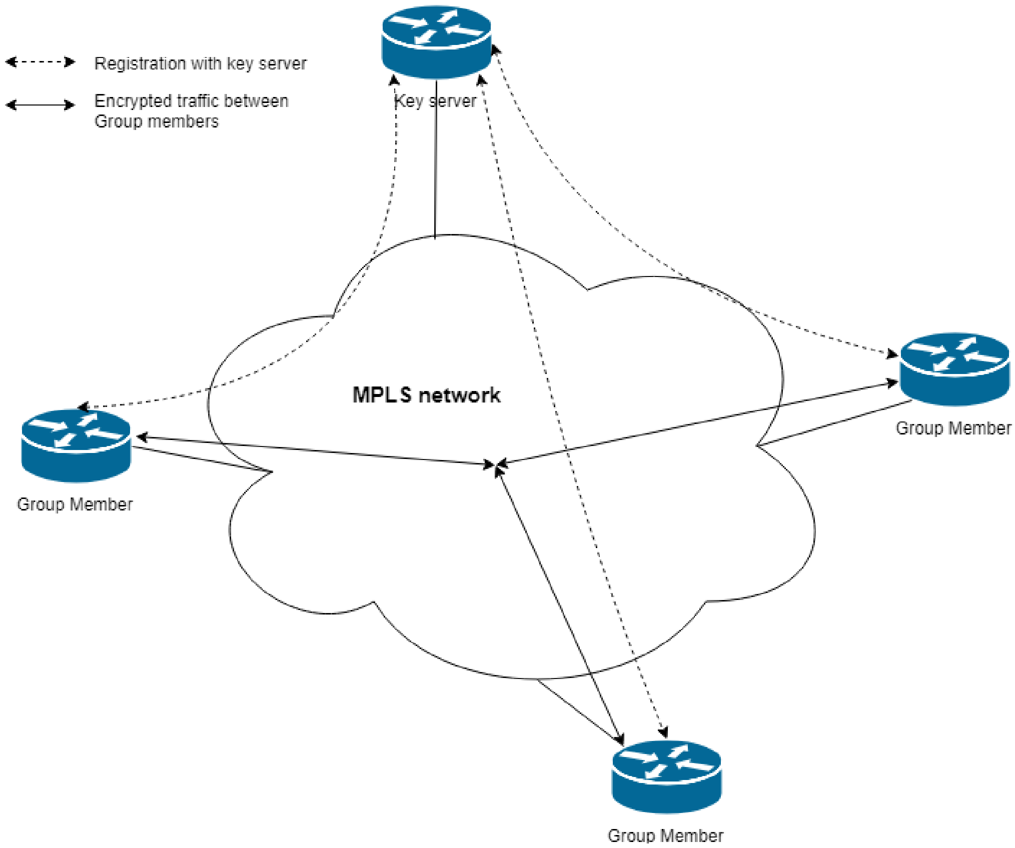

Registration with key server
Encrypted traffic between Group members
Key server
MPLS network
Group Member
Group Member
Group Member

Figure 3.3. *GETVPN architecture*

Initialization of shared SA starts when all GMs authenticate themselves with the Key server (KS) using IKE Phase 1 with either PSK or PKI. After GM has authenticated, it downloads the encryption policies and keys of the shared SA from the KS. Periodically KS pushes new keys and pseudotime to all GMs. Keys are used for encrypting the traffic and pseudotime is used for replay protection: all sent packets contain a timestamp of pseudotime and receiver compares the timestamp to the pseudotime it has. If a packet arrives too late, it is dropped.

In addition to using shared SA, GET VPN preserves the tunnel header. In traditional IPSec setups tunnel endpoint addresses are used as the source and the destination of the encrypted packet. In GETVPN, the original source and destination addresses are encapsulated to an outer header which allows packets to be routed using the underlying network infrastructure. The preservation of tunnel header makes GETVPN to be very well suited for MPLS (Multiprotocol Label Switching), layer 2 (L2) and IP networks with end to end IP connectivity as packets can be routed in the network natively.

Cisco markets Easy VPN as a solution for easier site-to-site VPN deployments which also supports remote-access setups for teleworkers [15]. However, Cisco has already stopped developing and supporting their remote-access client software [24] as they are concentrating on their AnyConnect VPN client which supports SSLVPN and IKEv2 IPsec but doesn't support Easy VPN as Easy VPN is limited to IKEv1. Therefore, users need to use 3rd party clients like Shrew [80] or TheGreenBow [98] if they want to keep using Easy VPN as remote-access solution.

As support for remote-access usage is limited, Easy VPNs real benefits are easier deployment and management compared to traditional point-to-point IPsec tunnels and support of hub-to-spoke architectures. Easy VPN uses Cisco Unity framework to which is their proprietary mechanism for defining and relaying IPsec parameters. Client device initiates a connection to an Easy VPN server and provides authentication credentials. The Easy VPN server, which is a hub in the architecture, then checks the provided credentials and pushes IPsec configuration to the client device. IPsec configuration is identical to all clients/spokes which makes management easier and centralized. Tunnels can be built automatically, manually or by traffic triggers, which means a tunnel is built when something tries to send traffic to the destination.

The primary offering from Cisco to remote access VPN solution market today is their proprietary AnyConnect. AnyConnect itself is the client software which is available for all major operating systems and mobile devices and VPN tunnels used by AnyConnect can be terminated with most of the modern Cisco enterprise routers and firewalls. Tunnels can either be SSL tunnels (TLS 1.2 and DTLS) or IPsec IKEv2 tunnels:

- DTLS is optimized for traffic requiring low latency, such as VoIP traffic.
- IPsec IKEv2 can be used when low latency is required and IPsec is required by security policies. [12]

3.1.2 Juniper

Juniper Networks, Inc. is an American networking and cybersecurity company, which develops and sells networking hardware and software and cybersecurity solutions. Juniper doesn't have SSL VPN in their offering any longer as they moved their Junos Pulse software and products to Pulse Secure in 2015 [45], but they offer IPsec based remote access VPN solution together with NCP engineering GmbH [77]. For site-to-site use cases, their routers and firewalls support IPsec and three proprietary VPN solutions called AutoVPN, Auto Discovery VPN (ADVPN) and Group VPN [9, 33].

AutoVPN is a hub-and-spoke, IPsec based solution which, like Cisco's DMPVN, consists of a central hub and multiple spokes. Advantage compared to basic site-to-site IPsec tunnel is that additional spokes can be added to the network without making any changes to the hub. Routing and IPsec related settings are only managed in the hub. These two things make the initial deployment and later the management easier. [42]

The underlying technology in AutoVPN is called next-hop tunnel binding (NHTB) [42]. NHTB allows multiple IPsec tunnels to be bound on the same logical tunnel interface, which simplifies the configuration as only a single interface is required in the hub [39]. Routing decisions are made based on the NHTB table, which maps destination IP prefixes to the next hop address which is the address of the VPN spoke. Hub-and-spokes configuration with NHTB and NHTB table are described in the picture 3.4 and the table 3.1.

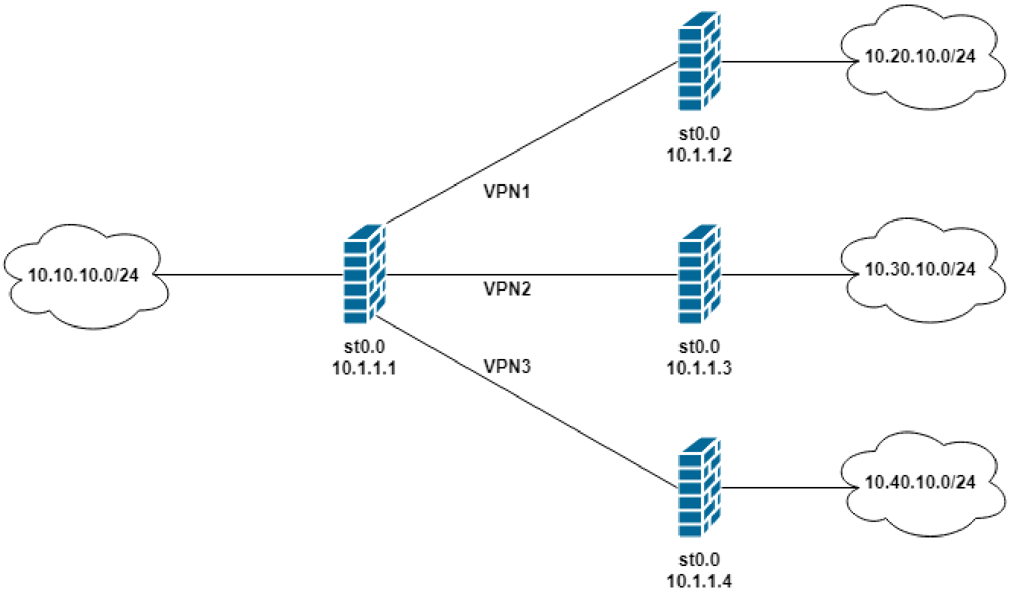


Figure 3.4. *Hub-and-spokes topology with NHTB*

Table 3.1. *Routing and NHTB tables*

Route Table			NHTB			
IP Prefix	Next Hop	Interface	Next Hop	Interface	IPsec VPN	Flag
10.20.10.0/24	10.1.1.2	st0.0	10.1.1.2	st0.0	VPN1	static
10.30.10.0/24	10.1.1.3	st0.0	10.1.1.3	st0.0	VPN2	static
10.40.10.0/24	10.1.1.4	st0.0	10.1.1.4	st0.0	VPN3	static

In the above example, 10.1.1.1 is the hub and other three devices are spokes. The routing table can be populated by using a dynamic routing protocol or static routes can be configured manually. NHTB can be populated by manually binding next-hop addresses to the VPN tunnels or hub can automatically get the next-hop addresses from the remote peer if both spoke and hub are Juniper devices. [39] If AutoVPN is used, manual binding of next-hop addresses is not supported [42].

AutoVPN doesn't support pre-shared keys, so only supported authentication method are

X.509 certificates, which require public key infrastructure (PKI) [42].

ADVPN is Juniper's solution for dynamical VPN tunnel creation. Base architecture is based on a hub and spokes: all spokes are connected to the central hub with static IPsec tunnels and spokes can communicate with each other via the hub. The tunnels between hub and spokes can be built with earlier mentioned AutoVPN. If needed, spokes can establish direct tunnels between each other using ADVPN. [9]

ADVPN protocol uses IKEv2 extension to transmit the required details to establish shortcut tunnels and devices supporting ADVPN extension advertise itself with ADVPN_SUPPORTED message in IKEv2 notify payload [9].

The shortcut establishment requires that both hub and spokes support ADVPN. Hub acts a shortcut suggester, which means it can suggest that two spokes, which can also be called shortcut partners, establish a direct shortcut tunnel between them. The shortcut suggester starts the process if it notices two spokes are trying to communicate with each other via the hub and they both have advertised being capable to ADVPN. [9]

During the shortcut exchange, the suggester sends suggestion first to one of the peers using the existing IKEv2 SA between the suggester and the partner. If the peer accepts exchange, the suggester starts exchange with the second peer. Shortcut exchange messages contain information which allows the peers to establish IKE and IPsec SA with each other. [9]

The shortcut suggester selects on of the peers as initiator and second one of the peers as responder. If neither of the peers is behind Network Address Translation (NAT), initiator is randomly selected. If one of the peers is behind NAT, it is selected as initiator. If both of the peers are behind NAT, shortcut cannot be established. The shortcut suggester starts the shortcut exchange with the responder and after the responder has accepted the suggestion, the suggester notifies the initiator. The initiator starts IKEv2 exchange with the responder using the details it gets from the suggester and establishes a new IPsec SA between the partners. After an IPsec SA has been successfully established, traffic starts to flow over the shortcut tunnel. [9]

Shortcuts don't have any pre-determined lifetime, but they are terminated if the amount of traffic falls below a specified rate for a specified time. When shortcut is terminated, the IKE SA and IPsec SAs are terminated, shortcut route is removed from both of the partners and traffic between them is then routed via the hub. [9]

In addition to being locked to Juniper devices only, ADVPN can only use X.509 certificates for authentication like AutoVPN [9]. In some cases, this can make the deployment of ADVPN infeasible.

For internal and MPLS networks, Juniper offers Group VPN, which is a set of features which allows using IPsec in an environment which requires multicast support or use of shared encryption keys. Like Cisco GET VPN, it is based on GDOI. Members of the group contact group server and authenticate themselves with IKE Phase 1 authentication. Then, members can retrieve shared keys and group SAs from the server and start communicating with other members via these shared SAs. If multicast is required, multicast packets are replicated in the core network like any other cleartext multicast packets but they are encrypted with a shared key. [33]

As both of Cisco and Juniper use the GDOI in their VPN solutions, they both share a limitation: GDOI doesn't encapsulate source and destination IP addresses. This means that the internal IP network addressing is exposed, but usually this is not an issue, as Group VPN should only be used in internal or MPLS network. [5]

Juniper has two versions of the Group VPN: v1 and v2. V1 is based on the GDOI defined in RFC 3547 and v2 is based on the updated version of GDOI defined in RFC 6407 [33, 34]. They are interoperable with some limitations, but v2 is only supported by the current generation of Juniper devices [33, 34]. V1 has some proprietary limitations so it might not fully work with GDOI implementations of other vendors, but v2 is fully compliant with the newer version of GDOI and Juniper states it interoperates with other RFC 6407 compliant devices [34].

Juniper's remote access VPN solution is different compared to some other remote access VPN solutions in the market. Unlike many others, Juniper doesn't have a SSL VPN available at all and they don't offer their own VPN client. Client, called NCP Remote Access Client, is provided by their partner NCP engineering GmbH and in addition to a Juniper SRX series device, the VPN solution requires a NCP Exclusive Remote Access Management server [77].

Remote access VPN supports IKEv1 and IKEv2. If the client is behind a firewall which blocks IKE traffic (UDP port 500) and prevents traffic required for setting up an IPsec tunnel, VPN endpoint and client can encapsulate the traffic and use TCP port 443 for the traffic. NCP calls this technology NCP Path Finder and they have versions 1 and 2 of it. Version 1 encapsulates the IPsec messages within a TCP connection over port 443. Version 2 encapsulates the messages within a SSL/TLS connection. Version 2 is used if it is supported and configured and if the client can't establish the connection with version 1. [77]

3.1.3 Pulse Secure

Pulse Secure is an American cybersecurity company which was born in 2015 when Juniper Networks sold Junos Pulse product line to Siris Capital [45]. Their products are mostly concentrated on securing remote access and the Pulse Connect Secure VPN is one part of that portfolio.

Pulse Connect supports both client and clientless modes and the client application is available to all major mobile and desktop operating systems. For securing the traffic it can use either TLS or IPsec with IKEv2 and both IPv4 and IPv6 are supported [74, 75]. As Pulse Secure is not building firewall devices, Pulse Connect server is always a dedicated VPN gateway device or virtualized appliance.

3.1.4 Checkpoint

Check Point Software Technologies Ltd. is a cybersecurity company from Israel. Check Point develops and sells software and hardware products for network, endpoint, cloud and mobile security. Their networking devices support normal IPsec for site-to-site VPN use cases and solution for remote access usage.

Check Point IPsec implementation has a feature called VPN Communities to ease the deployment of full mesh and star (hub and spoke) topologies [83]. To use VPN Communities, all participating devices need to have PSK or certificates as they are required for authentication before VPNs can be created and all devices need to be connected to Check Point Security Management Server. When all participating devices are connected to the management server, adding new peers to a new or existing IPsec star or mesh topology is relatively easy. [11, 82]

Remote access VPN of Check Point is like many others. It can use TLS or IPsec for encrypting the traffic, it supports both client and clientless modes and clients are available to all major operating systems [10]. VPN connections are terminated on Check Point firewalls but using the feature requires separate license.

3.1.5 Fortinet

Fortinet is a multinational company which headquarters are in California. They develop cybersecurity software, devices and services. Their FortiGate firewalls support normal IPsec, Fortinet proprietary remote access VPN, Overlay Controller VPN (OCVPN) and Auto Discovery VPN (ADVPN) which is similar to Juniper's ADVPN but not compatible with it.

OCVPN uses Fortinet cloud to simplify configuration of a VPN. Devices require license to use this feature and they must to have public IPs when they connect to the cloud. Only configuration OCVPN needs is the list of subnets which should be shared with the OCVPN participants and that OCVPN is enabled on all participants. After those have been configured, OCVPN firewalls send the data to the cloud and required configuration is generated in the cloud [27]. OCVPN can create single tunnel between two devices or if there are more than two participants, it can create full mesh topology or a hub-spoke topology with ADVPN shortcuts [27, 28].

ADVPN works similar to Juniper's ADVPN which was described in section 3.1.2. Hub(s), spokes and tunnels between them are configured manually or with the OCVPN. If spoke needs to send traffic to another spoke, it is routed through hub(s) to the receiving spoke.

When hub notices that both source and destination tunnels have auto-discovery enabled, it sends a message via IKE to the source and inform it should try to negotiate a direct tunnel for the traffic. [43]

The source spoke then sends a FortiOS-specific IKE INFORMATIONAL SHORTCUT-QUERY with public IP of the spoke, the source address of the traffic, the destination address of the traffic and the PSK towards the destination spoke via the hub(s). When the destination spoke receives the message, it replies with IKE INFORMATIONAL message containing the public IP of itself. [25]

When the source spoke has received the public IP of the destination spoke, it creates a new dynamic tunnel between the spokes and starts IKE negotiation. After successful tunnel creation routing is updated by the routing protocol (RIP, BGP or OSPF) traffic starts to flow via the shortcut. [25]

Remote access VPN is like many others. It supports both IPsec and SSL, client and clientless mode and FortiClient client software is available to all major operating systems.

3.1.6 Palo Alto Networks

Palo Alto Networks, Inc. is multinational cybersecurity company which has headquarters in California. The core of their product portfolio consists of next-generation firewalls and cloud-based cybersecurity services and products. Their firewalls support three different VPN solutions: basic IPsec for site-to-site connections, GlobalProtect VPN for remote access and GlobalProtect Large Scale VPN (LSVPN) for deploying a scalable hub and spoke solution [69].

The idea behind LSVPN is to simplify the deployment of a hub and multiple spokes, which are also called remote satellites, for enterprises. LSVPN requires Palo Alto firewalls to hub and each spokes but according to Palo Alto, the amount of required configuration is minimal. [68]

LSVPN uses certificates for authenticating the devices, SSL to encrypt the control and configuration communications a between hub and spokes and IPsec to encrypt the exchanged traffic between hub and spokes [69]. LSVPN supports both IPv4 and IPv6 protocols and static and dynamic routing. LSVPN deployment has three core components: GlobalProtect Portal, Gateway(s) and Satellite(s).

- Portal is the management component of the LSVPN infrastructure. Every spoke device connects to portal and downloads required configuration files to connect to the hub(s). [68]
- Gateway (a hub) acts as a tunnel endpoint for the satellite (spoke) connections. Portal and Gateway can be running a same Palo Alto firewall. [68]
- Satellite (spoke) is a firewall device on remote site that connect to the gateway with credentials and configurations downloaded from the portal. As configuration is

downloaded from portal, each new satellite requires very small amount of configuration before deploying. [68]

GlobalProtect VPN is Palo Alto's proprietary solution for remote access VPN. All of their firewalls support it but using it requires separate subscription. GlobalProtect has client software for all major mobile and desktop operating systems [67] and it also supports clientless mode for providing secure remote access to supported enterprise web applications [70]. Clientless mode uses SSL to encrypt the traffic.

In client mode, GlobalProtect supports both IPsec and SSL VPN based connections. IPsec is a preferred method as it has better performance than the SSL [109].

3.2 Open source solutions

The market of open source VPN solutions is dominated by two protocols: OpenVPN and IPsec. IPsec is usually used for site-to-site connections and the implementations should be compatible with commercial solutions. OpenVPN is an open source alternative for all commercial SSL VPNs and like commercial products, it has its own proprietary protocol.

3.2.1 OpenVPN

OpenVPN is open source VPN software which has been available since 2001. The original version was written by James Yonan and later he founded OpenVPN Inc. which has continued the development work since that. The community edition of OpenVPN is still free and open source but OpenVPN Inc. also sells a product called OpenVPN Access Server for business customers. Both community and commercial versions use the same OpenVPN protocol and offer the same core functionalities, but commercial version has some additional features like a web portal for administration, support for additional authentication methods like Lightweight Directory Access Protocol (LDAP) directory, easier installation and more options for access control [63]. OpenVPN can be considered as an industry standard in the market of open source VPN software as it is the most widely used open source VPN solution. Its status is so strong that IANA has given it an official assigned port 1194, which is the default port of the current releases.

OpenVPN is a SSL/TLS VPN solution which can be used as remote access and site-to-site VPN. It has own protocol, so it is not compatible with other open source or commercial solutions. It can use both UDP and TCP protocols and IPv4 and IPv6. OpenVPN supports two kinds of network adapters via the TUN/TAP driver: when using TUN, it creates layer 3 tunnel for IP traffic and with TAP it creates layer 2 tunnel for carrying Ethernet traffic. All encryption and authentication tasks are handled by open source OpenSSL and PolarSSL libraries which both are actively developed and patched. [65]

For authentication OpenVPN has two modes: static keys and TLS. If static keys are used, both peers have static keys which contain four separate keys: HMAC send, HMAC receive, encryption and decryption keys. In TLS mode, peers establish a SSL session and authenticate to each other with certificates. After successful authentication, both peers randomly generate encryption/decryption key and HMAC key source material and exchange it over the established SSL session. After the exchange, both peers have required keys to start exchanging encrypted traffic. [64]

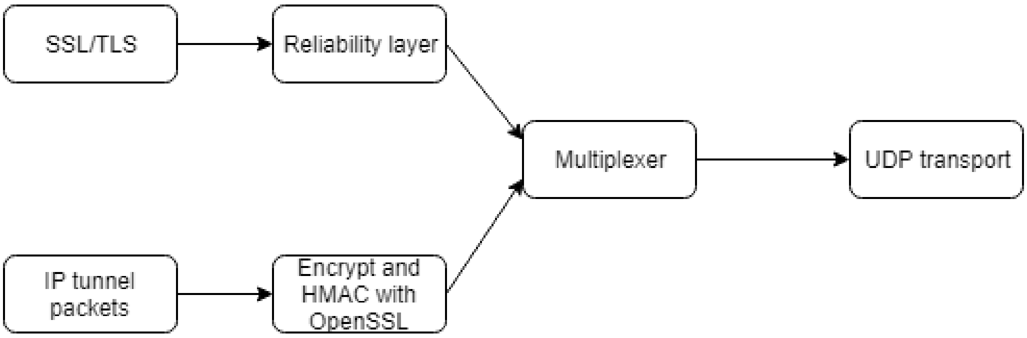


Figure 3.5. *OpenVPN multiplex model*

As SSL/TLS requires reliable transport, so OpenVPN has to offer a reliable transport to it even when using UDP with a separate reliability layer. OpenVPN multiplexes the SSL/TLS session and the encrypted packets to a single data stream, but only SSL/TLS session uses the reliability layer as the encrypted data stream doesn't require it. The multiplexing model is illustrated in the picture 3.5. [64]

Like many other VPN solutions, OpenVPN supports multiple platforms including Linux, Windows, OSX and mobile operating systems and it can be ported to any platform with relatively small effort as it has been created as a user-space daemon instead of a kernel module, encryption is handled by open source libraries and the architecture is modular. Being a user-space daemon also makes it more secure in the OS side than kernel implementations like IPsec as it can't access the most critical parts of the OS [36]. OpenVPN is also supported by open source firewall distributions like pfSense and VyOS.

3.2.2 IPsec

IPsec software is available for most of the operating systems in one or multiple forms. Most which are in active development implement the required standards mentioned in IPsec RFCs as defined in them.

For Linux distributions strongSwan and Libreswan are probably the most widely used software implementations. They are both descendants of the FreeS/WAN project which ended in 2004 and was forked to Openswan and strongSwan projects. Openswan was later forked to Libreswan. Both of them fully support IPsec but Libreswan has worse support for Extensible Authentication Protocol (EAP) and it can't be deployed in to a high available cluster unlike strongSwan [54, 94, 95].

FreeBSD and OpenBSD are supported by strongSwan but they also have their own IPsec implementation based on the KAME project [94, 97]. BSD based firewalls like pfSense and VyOS therefore have native IPsec support [16, 102]. VyOS also has their own implementation of Cisco's DMVPN described in section 3.1.1 and it is compatible with it Cisco's implementation [103].

3.3 Comparison

Based on the data collected in 3, it is clear that IPsec is the industry standard solution for site-to-site VPN tunnels. All commercial and open source products are supporting it and in most cases they fulfill the standards described in RFCs related to IPsec. Most of the large vendors have proprietary solutions of some kind for the deployment and configuration of site-to-site VPN tunnels which are not compatible with the products of other vendors but they all use IPsec for securing the traffic.

Even though IPsec can be considered as standard, the implementations and supported features differ more or less between the vendors and this can cause issues when tunnels are being built. This has happened because IPsec standard has large amount of options and flexibility [26]. Comparison of the IPsec implementations between multiple vendors is difficult and there aren't any mandatory certification processes to verify that vendor has implemented IPsec properly. Earlier there was a Virtual Private Network Consortium (VPNC) which did testing and tried to increase the interoperability but the consortium ended in 2015 [101]. Only company which is doing certifications of some kind for IPsec implementation today is ICSA Labs but getting certification from them is optional so there are currently very few IPsec certified devices in their list [40].

The proprietary solutions of vendors try to solve the complexity of deployment but most of them have limitations of some kind, they get complex to manage in large deployments and users are locked in to a single vendor when they are using those solutions. A comparison of proprietary IPsec based VPNs is shown in the table 3.2.

The above table is a simplification of the features as it only shows the interpreted goals of each solution. From table and earlier descriptions of each technology it is easy to see that most of all examined vendors have tried to develop technologies which would help deploying IPsec to large environments in basic hub-and-spoke manner and three out of five of them have also developed way to deploy full mesh with dynamic shortcuts without configuring everything manually. It is easy to understand why these have been the goals: IPsec is complicated and deploying even a single tunnel manually can be a time consuming task so it is not easy to deploy or maintain large environment.

Vendors have been able to ease the deployment and maintenance with these technologies but they don't solve the issues of IPsec completely as they only work if all devices come from the same vendor. Therefore, their usability in real world is limited because they often cannot be utilized when tunnels between different organizations are built as

Table 3.2. *Comparison of proprietary VPN solutions*

Product	Tunnelless	Full mesh	Dynamic shortcuts	Easier provisioning
Cisco DMVPN		X	X	X
Cisco FlexVPN		X	X	X
Cisco GETVPN	X			
Cisco Easy VPN				X
Juniper AutoVPN				X
Juniper ADVPN		X	X	
Juniper Group VPN	X			
Check Point Communities		X		X
Fortinet OCVPN		X		X
Fortinet ADVPN		X	X	
Palo Alto LSVPN				X

devices are not always from the same vendor. IPsec implementations of cloud vendors like Amazon or Microsoft are neither compatible with these proprietary solutions, so if organization wants to connect cloud infrastructure to their network with proprietary VPN technology they need to install a virtual appliance in cloud and let it handle the VPN instead of using managed VPN endpoint service [17, 104].

Remote access VPNs are today a big part of offering for each cybersecurity/networking company who sells them because users need to be able to reach resources in internal networks securely from home and other remote locations. None of the companies or open source projects have created completely new protocol for VPNs but they all rely on SSL/TLS and/or IPsec even though they are not compatible with each other. All of the available remote access VPN solutions have been available so long that the core features are almost identical in practice. Therefore vendors have started to develop more advanced extras like better centralized management, single sign-on (SSO), always-on VPN and 3rd party integrations to keep up with the competition and to distinguish from the competitors [12, 67, 75].

From all of these examined solutions, OpenVPN stands out for two reasons: it is the only open-source SSL VPN available and compared to the commercial SSL VPNs it is unique as it can be used also for site-to-site connections. Even though OpenVPN can be used for site-to-site connections it has not even properly challenged IPsec in enterprise environments because IPsec is supported by all enterprise networking devices unlike OpenVPN and it still offers better performance than OpenVPN [50, 52].

4 FUTURE

The need for remote access and interconnecting remote locations to each other over the Internet is not decreasing in the future, so VPNs and other solutions which secure the traffic and endpoints will be needed more and more. However, the transition to cloud services from local infrastructure, new technologies and new concepts can disrupt the position of traditional remote access VPNs and IPsec.

4.1 SoftEther

SoftEther is open source VPN software which is an academic project from University of Tsukuba. SoftEther was developed as a part of Daiyuu Nobori's master thesis research in the university and in 2014 it was released as open source. Even though SoftEther was released several years ago, it seems it is not widely used outside of Asia, as there are very few research papers or other written resources in English.

Original SoftEther 1.0 was only a simple layer 2 VPN software which was developed by Nobori as a personal project to bypass the firewalls of university. SoftEther 1.0 was released in 2003 as freeware and Government of Japan also got interested about it: they thought it is a danger to computer security and commercial VPN solutions and therefore demanded Nobori to stop sharing it to public. Nobori didn't agree with the government but he had to stop sharing it as otherwise he could have lost his place in the University of Tsukuba. [3]

In 2004, Nobori and Mitsubishi Materials Corporation made a 10-year agreement of selling the SoftEther 1.0 as a commercial VPN product of Mitsubishi. During the 10-year agreement, Nobori wasn't allowed to sell the SoftEther 1.0 to customers directly, but he was allowed to keep developing it and share it as freeware or open source. The current "SoftEther VPN", which is a completely different product than the original SoftEther 1.0, is the result of that development. Binaries of the SoftEther VPN were released in 2013 and in 2014 it became open source under GPLv2 license and in 2019 license was switched to Apache License 2.0. [19, 84, 105]

Like mentioned earlier, original SoftEther 1.0 only supported its own ethernet over HTTPS protocol for VPN, but when Nobori found out there isn't a VPN server available which would support multiple available VPN protocols, he started to develop the current version of the SoftEther VPN server. Currently, SoftEther VPN supports SoftEther, IPsec, SSTP and OpenVPN protocols and all of the protocols can be used simultaneously which makes SoftEther unique server software. [62, 106]

Key technology behind SoftEther is the virtualization of ethernet. By using virtualized ethernet on layer 2, SoftEther is not limited to tunneling IP traffic and performance is also better than with VPN solutions encapsulating IP packets. SoftEther creates a virtual ethernet adapter to the client device and a virtual ethernet switch (called Virtual Hub) to the VPN server. These two virtualized components are connected to each other via a VPN session which is like a virtualized ethernet cable. Sessions uses TCP or UDP connections and they are encrypted with SSL encryption. [106]

Even though virtual hubs are called hubs, they work as physical ethernet switches instead of hubs: ethernet frames are exchanged between each connected VPN session according to the data in the forwarding database (FDB). Virtual hubs can be connected to physical ethernet interfaces of the VPN server via local bridges. This allows the VPN server to be connected to physical infrastructure and allows remote users to access resources running on physical infrastructure. Virtual hubs are used for all protocols SoftEther supports. [106]

If required, hubs can be interconnected with a cascading connection, which is like a virtualized ethernet cable between two switches. Cascading allows, for example, interconnection between multiple sites and then users and computers connected to each site can reach the resources of other sites. [106]

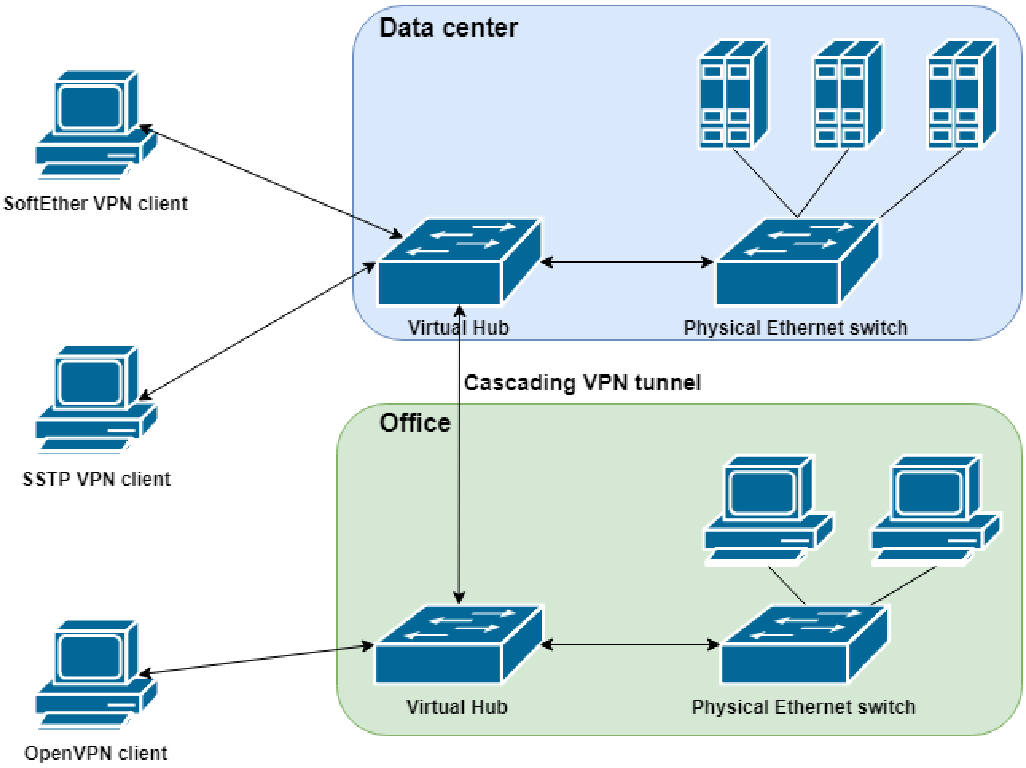


Figure 4.1. *SoftEther topology with cascading connection between office and data center*

Internally SoftEther server handles all VPN protocols on layer 2 even though OpenVPN, SSTP and IPsec use layer 3 for tunneling the traffic. SoftEther decrypts the arriving IP-packets and forwards them to a separate L2 <-> L3 converter which encapsulates the decrypted IP-packets to Ethernet frames. Ethernet frames are then sent to the hub and as the converter has unique IP- and MAC-address for each terminated VPN tunnel, the virtual hub is able to handle the incoming and outgoing traffic of the tunnel like all other native SoftEther traffic. [106]

The feature set of SoftEther is comparable to other open source and also commercial VPN products. It uses modern encryption algorithms to encrypt the traffic (AES-256 and 4098 bit RSA), it can bypass firewalls and NATs, it has sufficient logging capabilities, it supports RADIUS authentication and server and client software are available to all modern operating systems [106]. The firewall bypassing capabilities are somewhat unique: SoftEther can encapsulate VPN packets to ICMP or DNS packets. This feature enables SoftEther to be used in restricted networks which only permit ICMP (Internet Control Message Protocol) or DNS (Domain Name System) packets to be sent. [85]

4.2 WireGuard

WireGuard is a completely new point-to-point open source VPN application which uses state-of-the-art cryptography algorithms. WireGuard is not based on any old solution or protocol as it uses own protocol. According to the developers, WireGuard's goal is to be faster, simpler and more useful than old VPN solutions like IPsec and OpenVPN. As it is lightweight and fast, it can also be run on embedded and mobile devices. There isn't yet a stable release of the software but development is continuing and developers are working towards stable 1.0 release. [23]

The linux version of WireGuard has been created as a kernel module because the goal was to create fast and simple VPN solution which would be able to compete with IPsec in speed, could be run on slower hardware like mobile devices and would be easy to use, maintain and audit. Developers argue that these things have been achieved as performance is comparable to IPsec, implementation has less than 4000 lines of code so it is easy to audit and it is easy to use as WireGuard just creates virtual interfaces and integrates with native Linux tools like ip and ifconfig. [2]

As WireGuard aims to be easy to use, the authentication mechanism uses simple public and private keys for authentication. Mechanism resembles SSH closely as both server and client have public and private keys and only those need to be exchanged a before connection can be established. After a connection has been established with the help of public keys, key exchanges, connections, reconnects and disconnects are automatically handled by WireGuard. [2, 23]

Like IPsec and SSL VPNs, WireGuard operates on layer 3 so it can only be used for transmitting IP traffic. The authors of WireGuard argue that layer 3 is the correct way for bridging IP networks and that this approach is the cleaners for ensuring authenticity and attributability of the transmitted packets. It supports both IPv4 and IPv6 and can also encapsulate IPv4 to IPv6 and vice versa. [2]

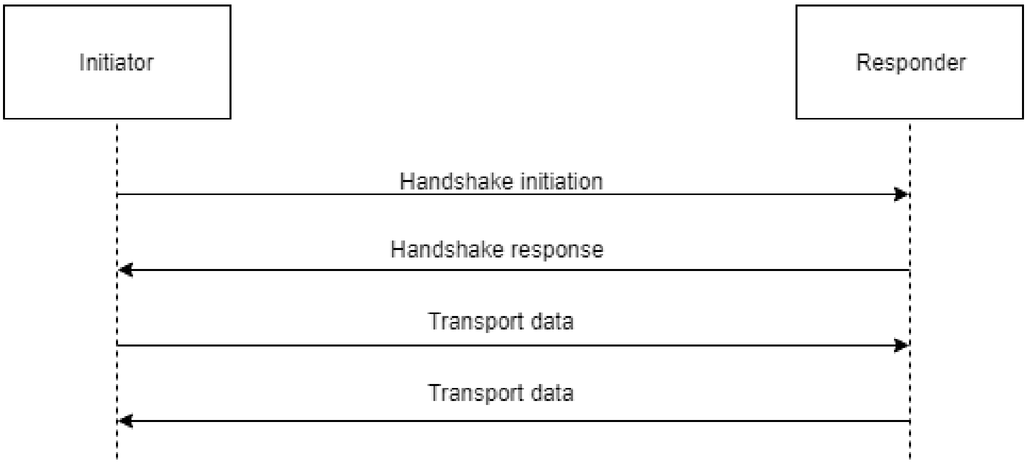


Figure 4.2. *Overview of handshake process in WireGuard when cookie is not used*

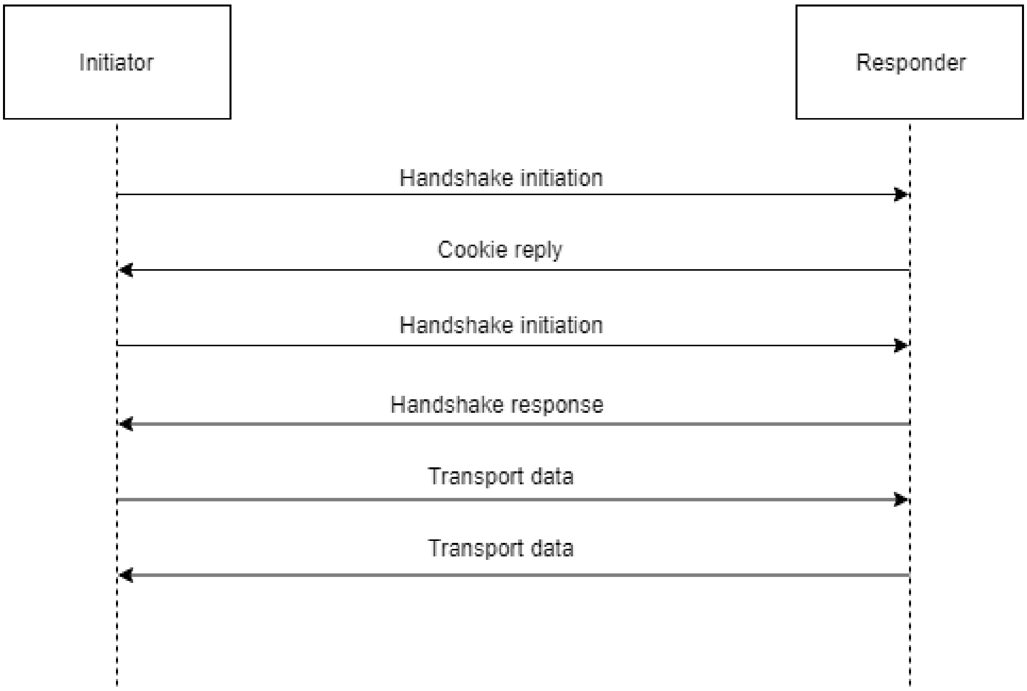


Figure 4.3. *Overview of handshake process in WireGuard when cookie is used*

Before any encrypted data can be sent, a handshake has to happen. In the handshake process, the initiator sends a handshake initiation message to the responder and the responder replies with a handshake response. Handshake messages are always authenticated with the public keys. After handshake messages have been exchanged, both peers compute the encryption keys and they can start transmitting encrypted data to each other. Handshake workflows with and without cookies are illustrated in pictures 4.2 and 4.3. [2]

WireGuard creates a simple virtual interface to the operating system. The interface associates tunnel IP address with the public key and the remote endpoint of the tunnel. If the interface tries to send a packet to the remote destination, workflow is following:

- Destination of the packet is 192.168.10.10.
- Address 192.168.10.10 is associated with peer ABCD
- Encrypt a packet with the public key of ABCD
- Send an encrypted packet to the peer's remote endpoint which address is 107.107.107.107 and UDP port is 53500.
- Peer ABCD receives a packet to 107.107.107.107:53500.
- The packet is decrypted with private key of the ABCD. The packet reveals it came from the peer EFGH and its current remote endpoint is 108.108.108.108:54500.
- Contents of the packet tell it came from 192.168.20.20. If configuration allows the interface to receive packets from EFGH from 192.168.20.20 it is accepted.

Creators of WireGuard call the above described process Cryptokey Routing. Cryptokey Routing is based on associating public keys with allowed IP addresses inside the tunnel. Each network interface has public and private keys and its own configuration. The configuration contains the private key of the interface, the public keys of the configured peers and their allowed IP addresses inside the tunnel. [2, 23]

When a client wants to send a packet, for example to 192.168.10.10, it uses the configuration as a routing table and searches for the destination address from the configuration and if the address is found, it uses the configured public key to encrypt the packet and sends it to the configured remote peer. On the server side, in other words on the receiving end of the tunnel, the list of allowed IPs works as an access control list: if a packet arrives from an IP which is not in the list of allowed, it is dropped. According to the developers, Cryptokey Routing eliminates the need for complicated firewalling as packets are only allowed to arrive from pre-defined and authenticated peers. [23]

From user perspective WireGuard seems to be like stateless. When the peer and the server have been configured properly, the client can start sending packets to the server immediately as WireGuard handles establishing the connection transparently. It also allows peers to roam from network to network as there aren't any active sessions or states to upkeep and peers keep updating the server with their current remote IP address. [23]

WireGuard doesn't utilize AES, 3DES or other widely used ciphers and protocols like IPsec and various SSL VPNs do. Instead of them, WireGuard is using other, newer available alternatives which have also been analyzed at level they can be considered secure. For initial key exchange, it uses Noise, encryption keys are computed with Curve25519 function, ChaCha20 with Poly1305 is used for authenticated symmetric encryption and BLAKE2s is used for hashing [2]. These protocols and ciphers have been selected because they offer better performance than the older alternatives. For example, ChaCha20 offers about 3 times better performance than AES if encryption is executed on platforms lacking specialized AES hardware [61].

Computing encryption keys with Curve25519 is CPU intensive so a denial of service attack would be possible if there wasn't any protection against it. WireGuard uses a cookie system to solve this potential security issue. If the receiving end of the package is under heavy CPU load, it can discard the incoming package and reply to the sender with a cookie reply containing a cookie. The sender can then re-transmit the package which was rejected with the cookie and get it accepted. The handshake process with cookie is shown in the picture 4.3 but cookies can also be utilized similar way with encrypted data packages. [2]

4.3 SDN and SD-WAN

Networking devices operate on three different planes:

- Management plane is used by the tools which are used to configure and manage the device. Usually, devices have graphical user interface and/or command line interface.
- Control plane controls the logic behind routing decisions.
- Data plane is responsible for transmitting the data packets according to the forwarding tables.

In traditional networking, management, control and data planes are on the same device which means that devices are specialized, complicated, expensive and each device has to be managed one by one. Software Defined Networking (SDN), which has been one of the hottest topics in last few years in networking, allows these planes to be separated from each other. When these three layers are separated from each other, control and management can be centralized which allows centralized management of multiple simple data plane devices. [32] SDN architecture is illustrated in the figure 4.4.

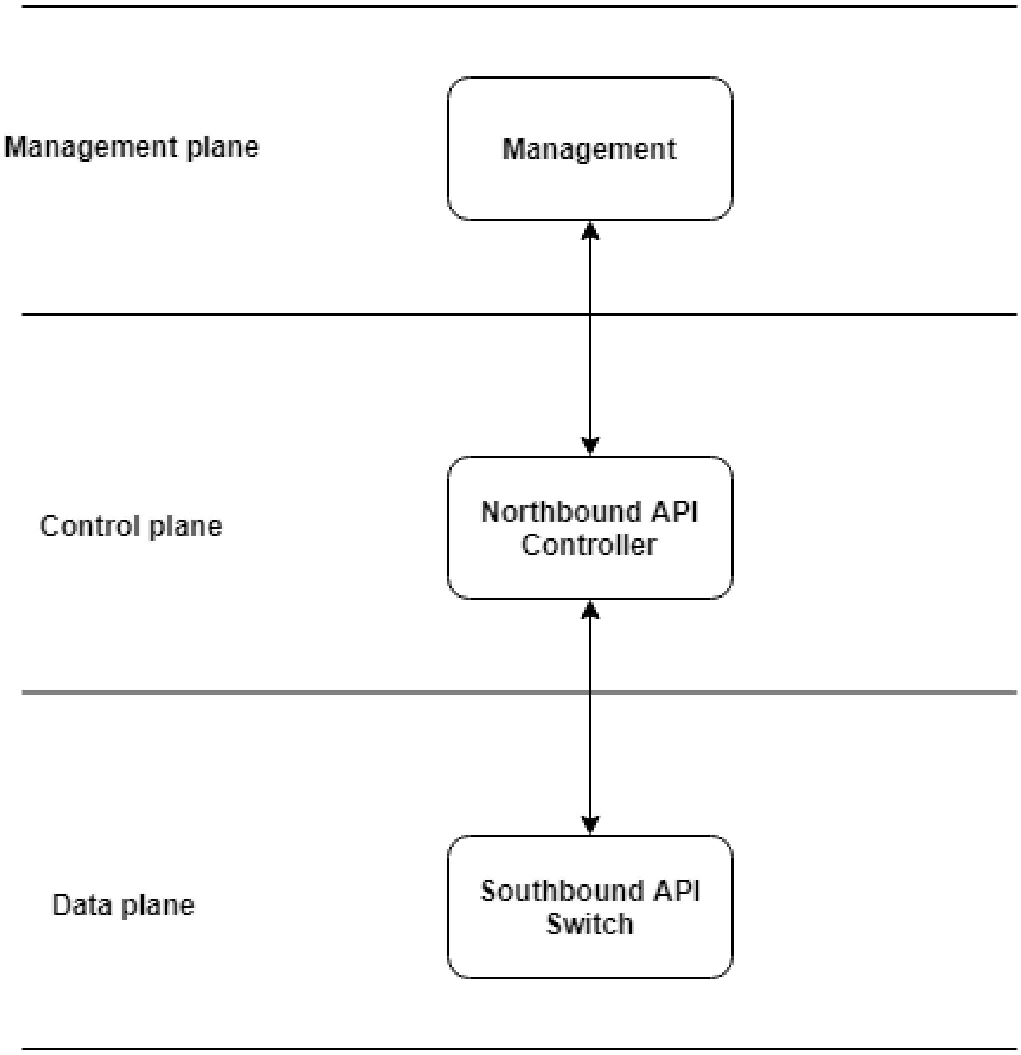


Figure 4.4. *SDN architecture*

Centralized management makes the configuration and management of the network easier and increases the network performance because the controller has visibility to the whole network and therefore it can make better routing decisions and utilize QoS more efficiently. Centralized management of data plane devices is done by controller software.

The crucial components of SDN architecture are northbound and southbound APIs. Northbound is the API between management and control planes and it can be used to monitor the traffic and control the controller software which allows the manipulation of traffic in the network. Southbound API is between control and data planes and controller software uses it to control the data plane devices. [32] The most widely utilized southbound protocol is an open standard called OpenFlow but networking vendors also have their proprietary solutions.

There are multiple controller software available. Most networking vendors who are selling SDN solutions have their own proprietary controllers but there are also multiple open source options available like ONOS, OpenDayLight and FloodLight.

Together with the rise of SDN, Software Defined Wide Are Networking (SD-WAN) concept has developed and increased its popularity. SD-WAN can thought as an extension of SDN, as SDN concept was originally designed and implemented into local area networks and SD-WAN spreads it to wide area networks. SD-WAN can be seen as replacement for old MPLS and traditional IPsec architectures as the centralized control makes the configuration and management significantly easier and faster. IPsec itself is still required with SDN and SD-WAN as they are not going to take away the need for encryption. In fact, encryption is required even more than before as SD-WAN relies on public Internet unlike private MPLS connections. It seems there aren't any new competitors to IPsec, so it is going to stay the most widely utilized encryption solutions.

Many of the large networking companies have their own SD-WAN solutions and there are also some completely new companies in the market as the market has significant growth potential. SD-WAN makes deployment and management of large networks with IPsec easier because the controller manages both ends of the tunnel, so SD-WAN solutions can replace the proprietary solutions like DMVPN and ADVPN which aimed to ease the configuration and management of IPsec. Compared to these old proprietary solutions, SD-WAN offers better compatibility and performance with cloud resources as traffic towards cloud can be routed directly to cloud, better scalability and better performance overall because controller can handle QoS better than traditional networking devices.

4.4 SDP

Software Defined Perimeter (SPD) is a relatively new concept which was published in 2013 by Software Defined Perimeter working group. The goal of the research group was to make a comprehensive and cost effective solutions for defeating attacks and securing users, cloud services and on-premises infrastructure. To achieve this all, they had three key elements for the design. [88]

- Identities of the users and devices must be verified before they can access the resources.
- Cryptographic verification is used to ensure that security model is followed.
- Protocols which are used must be proven public domain security controls.

SDP architecture consists of three components: SDP Client, SDP Controller and SDP Gateway. All components are shown in the figure 4.5. The both control and data connections of SDP are protected by TLS encryption. [88]

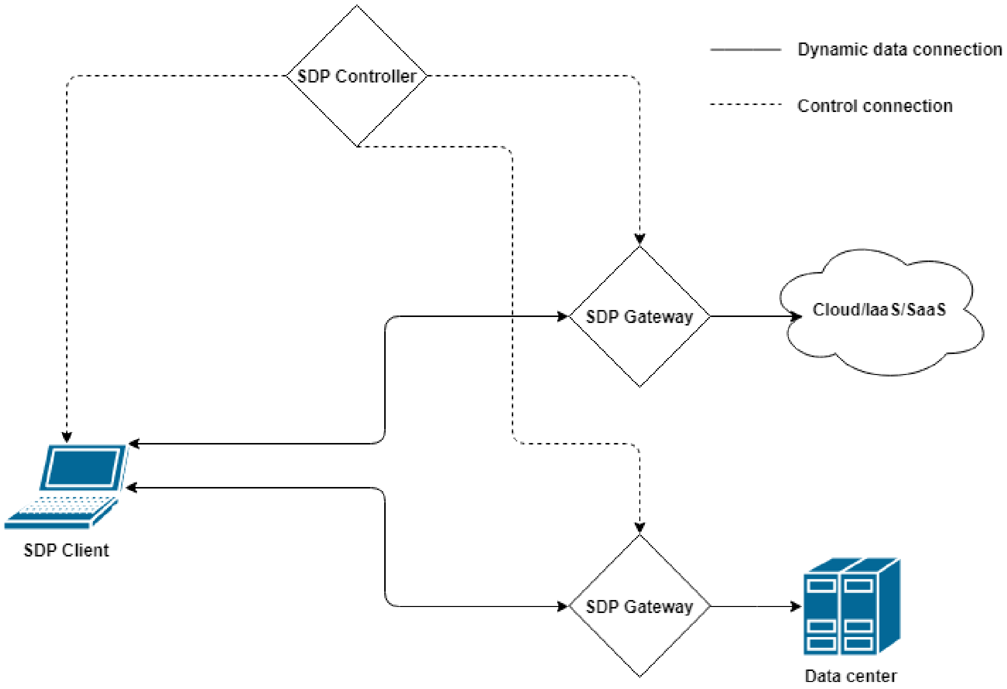


Figure 4.5. *SDP architecture*

The client initiates a session by connecting to the controller and attempts to authenticate the user and device. After a successful authentication, the controller verifies the client and devices are authorized to access the service they are trying to reach. If the client is authorized, the controller configures the gateway to accept connections from the client and configures the client to connect to the gateway. When the client is connecting, the gateway uses client's certificate and IP address to verify its identity and if they match those which controller provided, connection the requested resource is allowed. [88]

According to the research group, SDP has several unique security properties [88]:

- Only visible component of the architecture is the controller as none of the gateways or applications respond to anything which isn't authorized. This hides the complete application infrastructure.
- Pre-authentication and pre-authorization ensure that TCP connection between application and client is only initiated if client is allowed to access the application.
- Clients only have application level access and no network layer access.

Even though SDP concept is relatively new, there are already commercial solutions available from multiple vendors like Pulse Secure, Cato Networks and perimeter 81 [86, 87, 89]. As SDP has potential to be future solution, there will probably be even more competition in the market in the future.

5 DISCUSSION

This thesis examined the currently available commercial and open source VPN solutions and also some potential future technologies. The results of current situation are clear: IPsec is still industry standard solution for site-to-site connections and SSL/TLS is the most common technology utilized in remote access VPNs. They have almost completely replaced older technologies like PPTP and L2TP.

IPsec has so strong position that there are no real alternatives to it in most enterprise environments. This is interesting because IPsec is a relatively old standard, nothing revolutionary has been implemented into it in recent years and it has been criticized since the day one for being too complex, complicated and insecure [26]. None of those issues have been fixed but it is hard to see that IPsec would disappear anywhere in the near future.

OpenVPN has a strong position in the remote access VPN market and it is clearly the market leader of open source solutions. At least originally developers planned it would be a competitor to IPsec in site-to-site connections but that hasn't happened at least in the enterprise networks even though OpenVPN has almost as good performance as IPsec and it can be easier to manage than IPsec. The reason for that OpenVPN has not became more common in site-to-site connections is probably the fact that very few enterprise networking devices support it, so it is not an option for most of the enterprises. Still, OpenVPN has a really strong position in the enterprise and also in the consumer market as many of the commercial VPN providers use OpenVPN as their protocol.

From the new but traditional solutions, Wireguard looks more promising and interesting as SoftEther doesn't offer anything new except it supports multiple VPN technologies. SoftEther can be a good solution for environments where it can replace multiple VPN endpoints which utilize different technologies but it probably won't replace any other solutions in a large scale.

Wireguard has potential because it is modern new solution with new architecture and protocols and even though it is not ready yet, the results are promising. It is a lot more simple than IPsec and also easier to deploy and manage but it can still offer good security and performance. However, there aren't any research papers with performance measurements except the whitepaper of the Wireguard developer so the real performance has not been tested very well yet. Security of Wireguard is also still a question even though the protocol has been audited to be secure, but code itself has not been audited yet as it is still under development [23]. If code gets properly audited successfully and networking equipment start to support it or it gets a SDN compatible implementation, it might have a chance to become a real alternative to IPsec.

All of the above mentioned old and new solutions still share the same issues more or less: they all or just tunnels between two points so creating and managing a more complicated topology, like mesh, still requires quite large amount of work. The proprietary solutions like Cisco DMVPN and Juniper ADVPN and AutoVPN tried to ease the configuration and management but because they are proprietary and still somewhat clumsy, they haven't became very common. SDN solutions are the first ones which are really making the centralized management of networking infrastructure possible and therefore configuration and management of complex infrastructure can become easier. SDN architecture is neither completely proprietary like DMVPN and others, even though vendors also have proprietary solutions, and there are open protocols like OpenFlow which are supported by multiple vendors, which can make it more attractive. SDN is also more compatible with cloud services than traditional solutions. If SDN usage keeps increasing like it has done, the SDN solutions can completely supersede the old proprietary solutions like DMVPN.

The ever increasing usage of cloud based infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) services introduces new possibilities and also risks to the organizations which can be hard to handle with old solutions. For example, many cloud based services are available to public internet and their access control relies on user authentication, so VPNs don't offer any additional security to them. Even if cloud based service is protected by access control lists which only allow access through VPN, architecture is still heavily relying to the authentication of user who probably uses username, password and sometimes multi-factor authentication. Routing traffic to cloud through a centralized VPN server is also questionable performance wise.

The changed requirements of cloud based environments give chance to new innovations and approaches to the security issues. The SDP model is a great and promising example of that: it offers end-to-end encryption like traditional SSL/TLS VPN but without centralized VPN endpoint which makes it cloud compatible. It also offers additional integrated security features which traditional VPN solutions don't have: it hides the network completely from the outside attacker, strengthens the user and device authentication and limits possible exposure as users have only application level access to single applications instead of network level access to the infrastructure.

SDP solutions are also easier to maintain and manage to organizations as they are sold as SaaS service instead of appliances. When using SaaS service, organizations don't need to maintain the remote access solution by themselves and it is also scalable unlike traditional appliances.

Gartner did an analysis of SDP in 2018 and their message was that technical professionals should explore SDP as it matches the dynamic requirements of software defined data centers and clouds [44]. SDP solutions have a chance to gain significant share of the remote access market in the near future and replace traditional SSL/TLS VPNs. Especially companies which need to replace their current remote access solution in the near future might consider moving to SDP solution.

6 CONCLUSION

Virtual private networks have been utilized in information technology over 20 years and they are also needed in the future. During the over 20 years long period there have been relatively few technologies which would have been widely utilized. Standard IPsec and various SSL/TLS based are currently the two most widely used technologies in the VPNs deployed and they have been the most common choices for years. Nothing has replaced either of them as they have been good enough for data center environments.

The increasing usage of hybrid and cloud environments has changed the requirements for secure connectivity. Standard site-to-site IPsec and SSL/TLS based remote access VPNs are not very well suited for these new requirements. Standard IPsec is not scalable enough to large cloud environments and remote access VPNs which route all traffic through a data center are not optimal for using resources in multiple clouds.

SDN, SD-WAN and SDP answer these requirements with new features. SDN resolves some of the issues of IPsec by managing both ends of the tunnels which makes the configuration and management easier. SDP offers a completely new approach to the remote access which works well with cloud. Both SDN and SDP architectures are relatively new, but solutions based on them are already available from multiple vendors. It is highly likely that they are the future and their usage will increase fast in the near future.

Does IPsec have a future and will it be replaced with something new in the near future were two of the primary questions of this thesis. Answers are quite clear. IPsec has a future as it will still be used in the SDN networks. Wireguard might have a chance to be an altenative to IPsec but it probably won’t replace IPsec in a large scale, at least not very soon.

REFERENCES

[1] D. E. E. 3rd. *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 4305. RFC Editor, Dec. 2005, 1–9. URL: https://www.rfc-editor.org/rfc/rfc4305.txt.

[2] J. A. Donenfeld. WireGuard: Next Generation Kernel Network Tunnel. Jan. 2017. DOI: 10.14722/ndss.2017.23160.

[3] *About SoftEther VPN Project*. https://www.softether.org/9-about [Cited: 31.3.2019]. SoftEther Project.

[4] A. Alshamsi and T. Saito. A technical comparison of IPSec and SSL. *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*. Vol. 2. Mar. 2005, 395–398 vol.2. DOI: 10.1109/AINA.2005.70.

[5] J. Andersson. *Day One: IPsec VPN Cookbook 2018*. Juniper Networks Books, 2018. ISBN: 9781941441725.

[6] *ASA AnyConnect Double Authentication with Certificate Validation, Mapping, and Pre-Fill Configuration Guide*. 116111. https://www.cisco.com/c/en/us/support/docs/security/anyconnect-secure-mobility-client/116111-11611-config-double-authen-00.html [Cited: 3.7.2018]. Cisco Systems, Inc. June 2013.

[7] R. Atkinson. *IP Authentication Header*. RFC 1826. RFC Editor, Aug. 1995, 1–13. URL: https://www.rfc-editor.org/rfc/rfc1826.txt.

[8] R. Atkinson. *Security Architecture for the Internet Protocol*. RFC 1825. RFC Editor, Aug. 1995, 1–22. URL: https://www.rfc-editor.org/rfc/rfc1825.txt.

[9] *Auto Discovery VPNs*. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-auto-discovery-vpns.html [Cited: 13.1.2019]. Juniper Networks, Inc.

[10] *Check Point Remote Access Solutions*. https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk67820#Types%20of%20Remote%20Access%20Solutions%20-%20SSL%20VPN%20Portal%20for%20published%20business%20application [Cited: 5.5.2019]. Check Point Software Technologies Ltd.

[11] *Check Point VPN R76 Administration Guide*. https://sc1.checkpoint.com/documents/R76/CP_R76_VPN_AdminGuide/13894.htm [Cited: 5.5.2019]. Check Point Software Technologies Ltd.

[12] *Cisco AnyConnect Secure Mobility Client Data Sheet*. https://www.cisco.com/c/en/us/products/collateral/security/anyconnect-secure-mobility-client/datasheet-c78-733184.html [Cited: 13.1.2019]. Cisco Systems, Inc.

[13] *Cisco Cloud Services Router 1000V Series: IP Addressing: NHRP Configuration Guide*. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_nhrp/configuration/xe-16/nhrp-xe-16-book/config-nhrp.html [Cited: 29.9.2018]. Cisco Systems, Inc. Aug. 2016.

[14] *Cisco IOS DMVPN Overview*. https://www.cisco.com/c/dam/en/us/products/collateral/security/dynamic-multipoint-vpn-dmvpn/DMVPN_Overview.pdf [Cited: 29.9.2018]. Cisco Systems, Inc.

[15] *Cisco IOS Easy VPN*. hhttps://www.cisco.com/c/en/us/products/security/ios-easy-vpn/index.html [Cited: 29.12.2018]. Cisco Systems, Inc.

[16] *Configuring a Site-to-Site IPsec VPN*. https://docs.netgate.com/pfsense/en/latest/vpn/ipsec/configuring-a-site-to-site-ipsec-vpn.html [Cited: 5.5.2019]. Rubicon Communications LLC.

[17] *Create a Site-to-Site connection in the Azure portal*. https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-portal [Cited: 8.5.2019]. Microsoft Corporation.

[18] E. Crist and J. Keijser. *Mastering OpenVPN*. Packt Publishing, 2015. ISBN: 9781783553143. URL: https://books.google.fi/books?id=O-13CgAAQBAJ.

[19] *Declaration of license switch for SoftEther VPN from GPLv2 to Apache License 2.0*. https://www.softether.org/5-download/src/190121_switch_to_apache_license [Cited: 31.3.2019]. SoftEther Project.

[20] S. E. Deering and R. M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 1883. RFC Editor, Dec. 1995, 1–37. URL: https://www.rfc-editor.org/rfc/rfc1883.txt.

[21] J. P. Degabriele and K. G. Paterson. *Attacking the IPsec Standards in Encryption-only Configurations*. Cryptology ePrint Archive, Report 2007/125. https://eprint.iacr.org/2007/125. 2007.

[22] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, Aug. 2008, 1–104. URL: https://www.rfc-editor.org/rfc/rfc5246.txt.

[23] J. A. Donenfeld. *WireGuard homepage*. https://www.wireguard.com [Cited: 3.4.2019]. WireGuard.

[24] *End-of-Sale and End-of-Life Announcement for the Cisco VPN Client*. https://www.cisco.com/c/en/us/products/collateral/security/vpn-client/end_of_life_c51-680819.html [Cited: 29.12.2018]. Cisco Systems, Inc.

[25] *Example ADVPN configuration*. https://help.fortinet.com/fos60hlp/60/Content/FortiOS/fortigate-ipsecvpn/ADVPN/Config_Overview.htm [Cited: 5.5.2019]. Fortinet, Inc.

[26] N. Ferguson. A Cryptographic Evaluation of IPsec. (Nov. 2000).

[27] *FortiGate/FortiOS 6.2.0 Cookbook: Full mesh OCVPN*. https://docs.fortinet.com/document/fortigate/6.2.0/cookbook/850443/full-mesh-ocvpn [Cited: 5.5.2019]. Fortinet, Inc.

[28] *FortiGate/FortiOS 6.2.0 Cookbook: Hub-spoke OCVPN with ADVPN shortcut*. https://docs.fortinet.com/document/fortigate/6.2.0/cookbook/853223/hub-spoke-ocvpn-with-advpn-shortcut [Cited: 5.5.2019]. Fortinet, Inc.

[29] S. Frankel, S. Kelly and R. Glenn. *AES-CBC Cipher Algorithm Use with IPsec*. RFC 3602. RFC Editor, Sept. 2003, 1–15. URL: https://www.rfc-editor.org/rfc/rfc3602.txt.

[30] J. M. Galvin and K. McCloghrie. *Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC 1446. RFC Editor, Apr. 1993, 1–51. URL: https://www.rfc-editor.org/rfc/rfc1446.txt.

[31] *Global Protect 7.1 Administrator's Guide*. https://www.paloaltonetworks.com/documentation/71/globalprotect/globalprotect-admin-guide [Cited: 3.7.2018]. Palo Alto Networks, Inc.

[32] P. Göransson and C. Black. *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2014.

[33] *Group VPNv1*. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-group-vpnv1.html [Cited: 13.1.2019]. Juniper Networks, Inc.

[34] *Group VPNv2*. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-group-vpnv2.html [Cited: 27.1.2019]. Juniper Networks, Inc.

[35] K. Hamzeh, G. S. Pal, W. Verthein, J. Taarud, W. A. Little and G. Zorn. *Point-to-Point Tunneling Protocol (PPTP)*. RFC 2637. RFC Editor, July 1999, 1–57. URL: https://www.rfc-editor.org/rfc/rfc2637.txt.

[36] C. Hosner. *OpenVPN and SSL VPN Revolution*. SANS Institute, 2004.

[37] R. Housley. *Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)*. RFC 3686. RFC Editor, Jan. 2004, 1–19. URL: https://www.rfc-editor.org/rfc/rfc3686.txt.

[38] R. Housley. *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)*. RFC 4309. RFC Editor, Dec. 2005, 1–13. URL: https://www.rfc-editor.org/rfc/rfc4309.txt.

[39] *Hub-and-Spoke VPNs Using Next-Hop Tunnel Binding Overview*. https://www.juniper.net/documentation/en_US/release-independent/nce/topics/concept/vpn-hub-spoke-nhtb-example-overview.html [Cited: 3.2.2019]. Juniper Networks, Inc.

[40] *ICSA Labs*. https://www.icsalabs.com/ [Cited: 5.5.2019]. ICSA Labs.

[41] *IKEv2 Packet Exchange and Protocol Level Debugging*. 115936. https://www.cisco.com/c/en/us/support/docs/security-vpn/ipsec-negotiation-ike-protocols/115936-understanding-ikev2-packet-exch-debug.html [Cited: 18.6.2018]. Cisco. Mar. 2013.

[42] *Introduction to AutoVPN*. https://www.juniper.net/us/en/local/pdf/app-notes/3500214-en.pdf [Cited: 3.2.2019]. Juniper Networks, Inc.

[43] *IPsec Auto-Discovery VPN (ADVPN)*. https://help.fortinet.com/fos60hlp/60/Content/FortiOS/fortigate-ipsecvpn/ADVPN/ADVPN.htm?Highlight=advpn [Cited: 5.5.2019]. Fortinet, Inc.

[44] M. Judd and J. Fritsch. *Fact or Fiction: Are Software-Defined Perimeters Really the Next-Generation VPNs?*

[45] *Junos Pulse Products have Moved to a New Home*. https://www.juniper.net/us/en/pulsesecure/ [Cited: 13.1.2019]. Juniper Networks, Inc.

[46] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen and T. Kivinen. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. RFC Editor, Oct. 2014, 1–142. URL: https://www.rfc-editor.org/rfc/rfc7396.txt.

[47] S. Kent. *IP Authentication Header*. RFC 4302. RFC Editor, Dec. 2005, 1–34. URL: https://www.rfc-editor.org/rfc/rfc4302.txt.

[48] S. Kent. *IP Encapsulating Security Payload (ESP)*. RFC 4303. RFC Editor, Dec. 2005, 1–44. URL: https://www.rfc-editor.org/rfc/rfc4303.txt.

[49] S. Kent and K. Seo. *Security Architecture for the Internet Protocol*. RFC 4301. RFC Editor, Dec. 2005, 1–101. URL: https://www.rfc-editor.org/rfc/rfc4301.txt.

[50] I. Kotuliak, P. Rybár and P. Trúchly. Performance comparison of IPsec and TLS based VPN technologies. *2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. Oct. 2011, 217–221. DOI: 10.1109/ICETA.2011.6112567.

[51] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. Pearson, 2012. ISBN: 0132856204, 9780132856201.

[52] D. Lacković and M. Tomić. Performance analysis of virtualized VPN endpoints. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. May 2017, 466–471. DOI: 10.23919/MIPRO.2017.7973470.

[53] J. Lau, W. M. Townsley and I. Goyret. *Layer Two Tunneling Protocol - Version 3 (L2TPv3)*. RFC 3931. RFC Editor, May 2005, 1–94. URL: https://www.rfc-editor.org/rfc/rfc3931.txt.

[54] *Libreswan: IKEv2 CP and EAP support*. https://libreswan.org/wiki/IKEv2_CP_and_EAP_support [Cited: 5.5.2019]. Libreswan project.

[55] H. Lipmaa, D. Wagner and P. Rogaway. *Comments to NIST concerning AES modes of operation: CTR-mode encryption*. 2000.

[56] V. Manral. *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 4835. RFC Editor, Aug. 2007, 1–11. URL: https://www.rfc-editor.org/rfc/rfc4835.txt.

[57] M. Marlinspike. *Divide and Conquer: Cracking MS-CHAPv2 with a 100% success rate*. https://cookbook.fortinet.com/ssl-vpn-using-web-and-tunnel-mode-54/ [Cited: 30.7.2018]. July 2012.

[58] D. A. McGrew and J. Viega. *The Galois/Counter Mode of Operation (GCM)*. Submission to NIST. 2004.

[59] D. McGrew and P. Hoffman. *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 7321. RFC Editor, Aug. 2014, 1–11. URL: `https://www.rfc-editor.org/rfc/rfc7321.txt`.

[60] G. Meyer. *The PPP Encryption Control Protocol (ECP)*. RFC 1968. RFC Editor, June 1996, 1–11. URL: `https://www.rfc-editor.org/rfc/rfc1968.txt`.

[61] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. RFC 7359. RFC Editor, May 2015, 1–45. URL: `https://www.rfc-editor.org/rfc/rfc7539.txt`.

[62] D. Nobori. *Design and Implementation of SoftEther VPN*. `https://www.softether.org/@api/deki/files/399/=SoftEtherVPN.pdf` [Cited: 31.3.2019]. Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, Japan.

[63] *OpenVPN Access Server Datasheet*. `https://openvpn.net/datasheet/` [Cited: 14.4.2019]. OpenVPN Inc.

[64] *OpenVPN cryptographic layer*. `https://community.openvpn.net/openvpn/wiki/SecurityOverview` [Cited: 14.4.2019]. OpenVPN Inc.

[65] *Overview of OpenVPN*. `https://community.openvpn.net/openvpn/wiki/OverviewOfOpenvpn` [Cited: 14.4.2019]. OpenVPN Inc.

[66] G. S. Pall, B. Palter, A. Rubens, W. M. Townsley, A. J. Valencia and G. Zorn. *Layer Two Tunneling Protocol "L2TP"*. RFC 2661. RFC Editor, Aug. 1999, 1–80. URL: `https://www.rfc-editor.org/rfc/rfc2661.txt`.

[67] *Palo Alto Networks® Compatibility Matrix: What Features Does GlobalProtect Support?* `https://docs.paloaltonetworks.com/compatibility-matrix/globalprotect/what-features-does-globalprotect-support` [Cited: 4.5.2019]. Palo Alto Networks, Inc.

[68] *PAN-OS® Administrator's Guide: LSVPN Overview*. `https://docs.paloaltonetworks.com/pan-os/8-0/pan-os-admin/large-scale-vpn-lsvpn/lsvpn-overview.html` [Cited: 4.5.2019]. Palo Alto Networks, Inc.

[69] *PAN-OS® Administrator's Guide: VPN Deployments*. `https://docs.paloaltonetworks.com/pan-os/8-0/pan-os-admin/vpns/vpn-deployments` [Cited: 4.5.2019]. Palo Alto Networks, Inc.

[70] *PAN-OS® New Features Guide: Clientless VPN*. `https://docs.paloaltonetworks.com/pan-os/8-0/pan-os-new-features/globalprotect-features/clientless-vpn.html` [Cited: 4.5.2019]. Palo Alto Networks, Inc.

[71] B. V. Patel, B. Aboba, W. Dixon, G. Zorn and S. Booth. *Securing L2TP using IPsec*. RFC 3193. RFC Editor, May 2001, 1–28. URL: `https://www.rfc-editor.org/rfc/rfc3193.txt`.

[72] K. G. Paterson and A. K. Yau. *Cryptography in Theory and Practice: The Case of Encryption in IPsec*. Cryptology ePrint Archive, Report 2005/416. https://eprint.iacr.org/2005/416. 2005.

[73] R. Perlman and C. Kaufman. Key exchange in IPSec: analysis of IKE. *IEEE Internet Computing* 4.6 (Nov. 2000), 50–56. ISSN: 1089-7801. DOI: 10.1109/4236.895016.

[74] *Pulse Connect Secure Administrations Guide: IPv6 Support and Limitations for Pulse Connect Secure Features*. https://docs.pulsesecure.net/WebHelp/Content/PCS/PCS_AdminGuide_8.2/IPv6%20Support%20and%20Limitations.htm [Cited: 4.5.2019]. Pulse Secure LLC.

[75] *Pulse Connect Secure Datasheet*. https://www.pulsesecure.net/resource/pulse-connect-secure/ [Cited: 4.5.2019]. Pulse Secure LLC.

[76] D. Rand. *The PPP Compression Control Protocol (CCP)*. RFC 1962. RFC Editor, June 1996, 1–9. URL: https://www.rfc-editor.org/rfc/rfc1962.txt.

[77] *Remote Access VPNs with NCP Exclusive Remote Access Client*. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-remote-access-vpns-with-ncp-exclusive-remote-access-client.html [Cited: 7.2.2019]. Juniper Networks, Inc.

[78] B. Schneier and Mudge. Cryptanalysis of Microsoft's Point-to-point Tunneling Protocol (PPTP). *Proceedings of the 5th ACM Conference on Computer and Communications Security*. CCS '98. San Francisco, California, USA: ACM, 1998, 132–141. ISBN: 1-58113-007-4. DOI: 10.1145/288090.288119. URL: http://doi.acm.org/10.1145/288090.288119.

[79] B. Schneier, Mudge and D. Wagner. Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2). *Secure Networking — CQRE [Secure] ' 99*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, 192–203. ISBN: 978-3-540-46701-4.

[80] *Shrew Soft Support*. https://www.shrew.net/support/Main_Page [Cited: 29.12.2018]. Shrew Soft.

[81] W. A. Simpson. *PPP Authentication Protocols*. RFC 1334. RFC Editor, Oct. 1992, 1–16. URL: https://www.rfc-editor.org/rfc/rfc1334.txt.

[82] *Site to Site VPN R80.10 Administration Guide: Basic Site to Site VPN Configuration*. https://sc1.checkpoint.com/documents/R80.10/WebAdminGuides/EN/CP_R80.10_SitetoSiteVPN_AdminGuide/html_frameset.htm?topic=documents/R80.10/WebAdminGuides/EN/CP_R80.10_SitetoSiteVPN_AdminGuide/159310 [Cited: 5.5.2019]. Check Point Software Technologies Ltd.

[83] *SmartConsole R80 Help: VPN Communities - Gateways*. https://sc1.checkpoint.com/documents/R80/CP_R80_SmartDashboard_OLH/html_frameset.htm?topic=documents/R80/CP_R80_SmartDashboard_OLH/yTvgN9yEbFdKrDvx-9meng2 [Cited: 5.5.2019]. Check Point Software Technologies Ltd.

[84] *SoftEther VPN Becomes Open Source*. https://www.softether.org/9-about/News/800-open-source [Cited: 31.3.2019]. SoftEther Project.

[85] *SoftEther VPN Manual*. https://www.softether.org/4-docs/1-manual [Cited: 31.3.2019]. SoftEther Project.

[86] *Software Defined Perimeter Access for Hybrid IT*. https://www.pulsesecure.net/products/sdp-overview/ [Cited: 13.5.2019]. Pulse Secure LLC.

[87] *Software Defined Perimeter (SDP)*. https://www.catonetworks.com/solutions/software-defined-perimeter-sdp/ [Cited: 13.5.2019]. Cato Networks.

[88] *Software Defined Perimeter Working Group*. https://cloudsecurityalliance.org/working-groups/software-defined-perimeter/#_overview [Cited: 13.5.2019]. Cloud Security Alliance.

[89] *Software-Defined Perimeter*. https://www.perimeter81.com/solutions/software-defined-perimeter [Cited: 13.5.2019]. Safer Social Ltd.

[90] S. Song. *SSL VPN Security*. https://www.cisco.com/c/en/us/about/security-center/ssl-vpn-security.html [Cited: 3.7.2018]. Cisco Systems, Inc.

[91] *SSL VPN*. https://www.barracuda.com/glossary/ssl-vpn [Cited: 30.7.2018]. Barracuda Networks, Inc.

[92] *SSL VPN using web and tunnel mode*. https://cookbook.fortinet.com/ssl-vpn-using-web-and-tunnel-mode-54/ [Cited: 30.7.2018]. Fortinet, Inc. Dec. 2015.

[93] R. Stanton. Securing VPNs: comparing SSL and IPsec. *Computer Fraud Security* 2005.9 (2005), 17–19. ISSN: 1361-3723. DOI: https://doi.org/10.1016/S1361-3723(05)70254-2. URL: http://www.sciencedirect.com/science/article/pii/S1361372305702542.

[94] *strongSwan: the OpenSource IPsec-based VPN Solution*. https://www.strongswan.org [Cited: 5.5.2019]. strongSwan project.

[95] *strongSwan User Documentation: High Availability*. https://wiki.strongswan.org/projects/strongswan/wiki/HighAvailability [Cited: 5.5.2019]. strongSwan project.

[96] A. S. Tanenbaum and D. Wetherall. Computer networks, 5th Edition. 2011.

[97] *The KAME project*. http://www.kame.net [Cited: 5.5.2019]. The KAME project.

[98] *TheGreenBow Compatible VPN Gateways*. http://www.thegreenbow.com/vpn_gateway.html [Cited: 29.12.2018]. TheGreenBow.

[99] *Transport Layer Security Protocol*. https://docs.microsoft.com/en-us/windows/desktop/secauthn/transport-layer-security-protocol [Cited: 11.7.2018]. Microsoft.

[100] J. Viega and D. A. McGrew. *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)*. RFC 4106. RFC Editor, June 2005, 1–11. URL: https://www.rfc-editor.org/rfc/rfc4106.txt.

[101] *Virtual Private Network Consortium*. https://www.vpnc.org/ [Cited: 5.5.2019]. VPN Consortium.

[102] *VyOS User Guide: Site-to-Site IPsec*. https://wiki.vyos.net/wiki/User_Guide#Site-to-Site_IPsec [Cited: 5.5.2019]. vyos.io.

[103] *VyOS wiki: DMVPN*. https://wiki.vyos.net/wiki/DMVPN [Cited: 5.5.2019]. vyos.io.

[104] *What is AWS Site-to-Site VPN?* https://docs.aws.amazon.com/vpn/latest/s2svpn/VPC_VPN.html [Cited: 8.5.2019]. Amazon Web Services, Inc.

[105] *What is SoftEther VPN*. https://www.softether.org/#What_is_SoftEther_VPN [Cited: 31.3.2019]. SoftEther Project.

[106] *What is SoftEther VPN*. https://www.softether.org/ [Cited: 31.3.2019]. SoftEther Project.

[107] *What is the difference between the Tunnel and Transport modes in ESP?* KB5302. Version 5.0. Juniper Networks. Dec. 2017.

[108] D. Whiting, R. Housley and N. Ferguson. *Counter with CBC-MAC (CCM)*. RFC 3610. RFC Editor, Sept. 2003, 1–26. URL: https://www.rfc-editor.org/rfc/rfc3610.txt.

[109] *Why is GlobalProtect Slower on SSL VPN Compared to IPsec VPN?* https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000ClXPCA0 [Cited: 4.5.2019]. Palo Alto Networks, Inc.