

POLITECNICO DI TORINO

INFORMATION SYSTEMS SECURITY

A Comparative Study of IPsec, WireGuard, and OpenVPN: An Evaluation of End-to-End Encryption and Site-to-Site VPN Solutions

Author:

Pedro MORAIS

s307250@studenti.polito.it

February 8, 2023



**Politecnico
di Torino**

Contents

1	Introduction	3
1.1	Brief overview of end-to-end encryption and site-to-site VPN solutions	3
1.2	Importance of information security in today's digital world	3
1.3	Purpose and scope of the report	4
2	IPsec	5
2.1	Protocol and Features	5
2.2	Security Algorithms	6
2.3	Transport mode and Tunnelling mode	7
2.4	Advantages and disadvantages	9
3	WireGuard	9
3.1	Protocols and Features	10
3.2	Security algorithms	11
3.3	Design and Implementation	12
3.4	Advantages and disadvantages	12
4	OpenVPN	13
4.1	Protocols and Features	14
4.2	Security Algorithms	15
4.3	SSL/TLS based VPN	16
4.4	Advantages and Disadvantages	18
5	Theoretical comparison between IPsec, WireGuard, and OpenVPN	19
6	Practical Evaluation	22
6.1	Experimental Setup and Initial Configurations	22
6.1.1	Methodology and Test Design	22
6.1.2	Kali Linux and Virtual Box	23
6.1.3	Installing the Softwares	25
6.2	E2EE Tests	26
6.2.1	No Encryption (Benchmark)	26
6.2.2	AES-256 with HMAC-SHA256 authentication in IPsec	27
6.2.3	3DES with HMAC-SHA1 authentication in IPsec	28
6.2.4	Chacha20 with Poly1305 authentication in WireGuard	28
6.2.5	AES-256 with SHA-256 authentication in OpenVPN	30
6.2.6	Blowfish with SHA-384 authentication in OpenVPN	34
6.2.7	Results of the Evaluation	35

6.3	Site-to-site VPN Tests	37
6.3.1	No encryption (Benchmark test)	37
6.3.2	AES-256 with HMAC-SHA256 authentication in IPsec	38
6.3.3	3DES with HMAC-SHA1 authentication in IPsec	38
6.3.4	Chacha20 with Poly1305 authentication in WireGuard	39
6.3.5	AES-256 with SHA-256 authentication in OpenVPN	39
6.3.6	Blowfish with SHA-384 authentication in OpenVPN	39
6.3.7	Results of the Evaluation	40
7	Conclusions	42
7.1	Summary of the findings and results of the comparative evaluation	42
7.2	Recommendation for the most suitable solution based on specific requirements and use cases	43
7.3	Discussion of the future trends and directions for end-to-end encryption and site-to-site VPN solutions	44

List of Figures

1	AH + ESP Transport Mode [6]	7
2	AH + ESP Tunnel Mode [6]	8
3	IPsec Tunnel Mode [5]	8
4	The two modes OpenVPN offers [1]	15
5	Connection between a client and a server using OpenVPN [1]	16
6	Bridged network diagram [25]	25
7	E2EE test scheme	27
8	E2EE tests with 10 KB traffic	35
9	E2EE tests with 100 KB traffic	35
10	E2EE tests with 1 MB traffic	35
11	E2EE tests with 10 MB traffic	36
12	E2EE tests with 100 MB traffic	36
13	Site-to-Site VPN test scheme	37
14	Site-to-Site tests with 10 KB traffic	40
15	Site-to-Site tests with 100 KB traffic	40
16	Site-to-Site tests with 1 MB traffic	40
17	Site-to-Site tests with 10 MB traffic	41
18	Site-to-Site tests with 100 MB traffic	41

1 Introduction

1.1 Brief overview of end-to-end encryption and site-to-site VPN solutions

In today's fast-paced digital world, the protection and confidentiality of sensitive information is a critical concern for individuals and organizations alike. To ensure secure communication and data transfers, various encryption and virtual private network (VPN) technologies have been developed and widely adopted. End-to-end encryption and site-to-site VPN solutions are two such technologies that are widely used for securing data transmissions and providing privacy over public networks.

End-to-end encryption (E2EE) is a method of encrypting data so that only the communicating parties can access the information. The encryption is applied at the endpoints of the communication, ensuring that the data remains confidential and protected from unauthorized access as it travels over the internet. E2EE is commonly used to secure communication between devices, such as smartphones, laptops, or servers.

Site-to-Site VPN, on the other hand, is a method of securely connecting two or more private networks over the public internet. Site-to-Site VPN allows organizations to extend their private network to remote locations, providing secure and encrypted communication between their networks. This type of VPN is commonly used to connect branch offices, data centers, or cloud services to the main corporate network. Site-to-Site VPN uses encryption and authentication mechanisms to protect the confidentiality and integrity of the communication between the two private networks.

The main difference between E2EE and Site-to-Site VPN is the scope of the communication. E2EE is used to secure communication between two endpoints, such as between two devices or applications, while Site-to-Site VPN is used to secure communication between two or more private networks. Both methods are important for ensuring the security and privacy of communication over the internet, but their specific use cases and implementations differ based on the needs and requirements of the communication.

1.2 Importance of information security in today's digital world

The importance of information security in today's digital world is a multifaceted issue. With the rapid growth of technology and the widespread use of the internet, sensitive and confidential information is being transmitted and stored in huge quantities on a daily basis. This information includes personal and financial data, intellec-

tual property, business plans and strategies, and confidential communications. The protection of this information is of utmost importance as its unauthorized access, theft, or exploitation could result in significant harm to individuals, organizations, and even entire nations.

In addition to the threat of data breaches and theft, the rise of cyber attacks has added a new level of complexity to the issue of information security. Malware, phishing, and hacking are just some of the threats that organizations and individuals face when using the internet. These attacks can cause serious damage, including the loss of sensitive information, financial losses, and reputational harm. In the worst-case scenario, these attacks can result in the disruption of essential services and the compromise of national security.

To mitigate these risks, individuals and organizations must adopt robust and effective security measures to protect their information. End-to-end encryption and site-to-site VPN solutions play a critical role in ensuring the confidentiality and privacy of sensitive information during transmission and storage. By providing a secure and private communication channel, these solutions help to reduce the risk of unauthorized access, theft, and exploitation of information. Moreover, end-to-end encryption and site-to-site VPN solutions offer a level of security that goes beyond simple password protection, adding an extra layer of protection for sensitive information.

In short, the importance of information security in today's digital world cannot be overstated. With the increasing threat of cyber attacks and the growing volume of sensitive information being transmitted and stored online, it is imperative that individuals and organizations prioritize the protection of their information. End-to-end encryption and site-to-site VPN solutions play a critical role in ensuring the confidentiality and privacy of sensitive information, and should be considered essential components of any comprehensive security strategy.

1.3 Purpose and scope of the report

The purpose of this report is to provide a comprehensive overview of three widely used solutions for end-to-end encryption and site-to-site VPN: IPsec, WireGuard, and OpenVPN. The report aims to compare and contrast these solutions both theoretically and practically, providing a comprehensive evaluation of their features, protocols, algorithms, and performance.

The scope of this report covers the following aspects of each solution: an introduction to the technology, a detailed analysis of the underlying protocols and algorithms, a comparison of the features and capabilities, a discussion of the strengths and weaknesses of each solution, and an evaluation of the practical performance

based on experimental setup and measurements.

The report will be of interest to individuals and organizations looking to secure their sensitive information during transmission and storage. It will also be of value to security professionals, students, and researchers in the field of information security. The report provides a comprehensive evaluation of each solution, making it an indispensable resource for anyone looking to make informed decisions about the protection of their information.

Finally, the report concludes with a demonstration of the setup used for the experimental evaluation, allowing readers to gain hands-on experience with the solutions and understand how they can be implemented in real-world scenarios. The report aims to provide a comprehensive and detailed evaluation of the three solutions, serving as a valuable resource for anyone looking to secure their information in today's digital world.

2 IPsec

IPsec, short for Internet Protocol Security, has been in development since the late 1980s, with the first IPsec specification being published in 1998. The development of IPsec was driven by the need to provide secure communication over the internet, as the internet was rapidly growing and becoming more widely used. IPsec was initially designed as a protocol suite that would provide confidentiality, integrity, and authenticity to internet communications, protecting sensitive information from unauthorized access, theft, and exploitation. Over the years, IPsec has evolved to include new features and capabilities, making it one of the most widely used VPN solutions today.

2.1 Protocol and Features

The Internet Protocol Security is a set of security protocols that provide secure communications over an internet protocol (IP) network. The protocol suite is composed of two main protocols: the Internet Key Exchange (IKE) protocol and the Encapsulating Security Payload (ESP) or Authentication Header (AH) protocols.

The IKE protocol is used to establish a secure communication channel between two devices and is responsible for creating, managing, and exchanging the encryption and authentication keys. IKE uses a two-phase process, in which the first phase establishes a secure channel by negotiating the security parameters and the second phase uses that channel to establish the shared secret keys. The IKE protocol supports various authentication methods such as digital certificates, pre-shared keys, and biometric authentication.

The Encapsulating Security Payload and Authentication Header protocols are both used in IPsec to provide encryption and authentication for the data transmitted over an IP network. They are often used together to provide a combination of confidentiality, integrity, and authenticity.

ESP provides confidentiality by encrypting the data, and it also provides integrity and authenticity by adding an ESP header to the packet. The ESP header contains information such as the security association (SA) identifier and a sequence number, which are used for tracking and managing the security parameters of the connection. ESP can also provide anti-replay protection, which helps prevent replay attacks by checking if a packet has been received before.

AH provides authentication by adding an Authentication header to the packet, which includes a digest of the packet's contents, ensuring that the packet has not been tampered with during transit. AH does not provide confidentiality, as it doesn't encrypt the payload.

Together, ESP and AH provide a high level of security by encrypting the data to protect it from eavesdropping, and also providing authentication to protect against tampering and replay attacks. Organizations can choose to use ESP or AH individually depending on their security needs, or they can use them together for maximum protection.

Overall, IPsec provides a robust and secure solution for VPNs by implementing a combination of different protocols to ensure confidentiality, integrity, and authenticity of the data transmitted over an IP network.

2.2 Security Algorithms

The security algorithms used in IPsec provide the encryption and authentication that is necessary for securing the data transmitted over an IP network. These algorithms are used in both the IKE and ESP/AH protocols to ensure confidentiality, integrity, and authenticity of the data.

The most commonly used encryption algorithm in IPsec is the Advanced Encryption Standard (AES). AES is a symmetric key algorithm that can use 128, 192 or 256-bit keys. The longer the key, the stronger the encryption. AES is considered to be one of the most secure encryption algorithms available and is widely used in government and enterprise networks.

Other encryption algorithms that can be used in IPsec include Triple Data Encryption Standard (3DES) and Blowfish. 3DES is a symmetric key algorithm that uses a 168-bit key, which is considered to be less secure than AES. Blowfish is another symmetric key algorithm that uses a variable-length key, and it is considered to be secure, but less widely supported than AES.

Authentication algorithms used in IPsec include Secure Hash Algorithm 1 (SHA-1), Secure Hash Algorithm 256 (SHA-256), and Secure Hash Algorithm 512 (SHA-512). These algorithms are used to generate a unique, fixed-size value called a hash from the original data, which can then be used to verify the integrity of the data. SHA-256 and SHA-512 are considered more secure than SHA-1 and are recommended for use in new implementations.

In summary, IPsec provides a wide range of options for encryption and authentication algorithms to choose from, depending on the security requirements of the organization. AES is considered to be the most secure encryption algorithm while SHA-256 and SHA-512 are considered the most secure hash algorithm. It is also worth noting that the newer algorithms generally provide a higher level of security, but they may require more resources and computational power, so organizations should consider that when deciding which algorithms to use.

2.3 Transport mode and Tunneling mode

Transport mode and Tunneling mode are two different ways of implementing IPsec, which are used to secure different types of IP traffic.

Transport mode is used to provide end-to-end security for individual IP packets, by encrypting and authenticating the data at the IP level. In transport mode, the original IP header remains unchanged, and only the payload (i.e., the data portion of the packet) is encrypted and protected by ESP or AH protocol. This mode is typically used for E2EE, where the goal is to protect the communication between two endpoints.

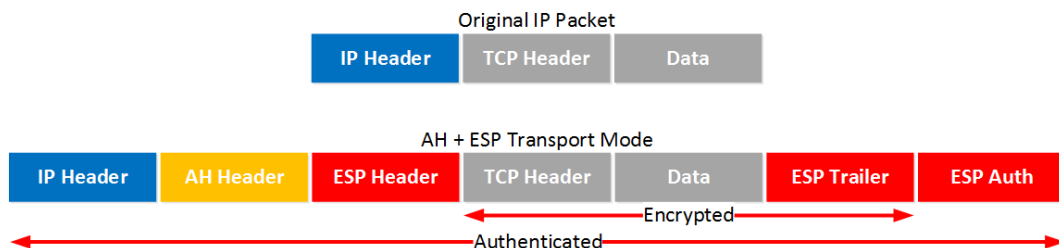


Figure 1: AH + ESP Transport Mode [6]

On the other hand, Tunneling mode is used to create site-to-site VPNs, by encrypting and authenticating the entire IP packet. In tunneling mode, a new IP header is added to the original packet, creating a new "tunneled" packet. This new packet is then encrypted and protected by ESP or AH protocol. The original IP header and payload are encapsulated within the new "tunneled" packet, and the new packet is then sent over the public network.

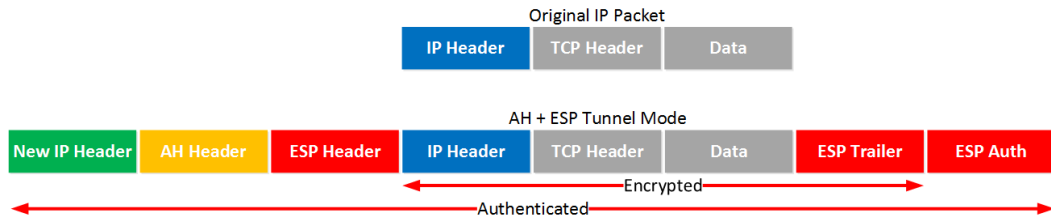


Figure 2: AH + ESP Tunnel Mode [6]

Building an IPsec Tunnel is a two-phase process. The first phase is known as the IKE Phase 1, which establishes a secure channel between the two devices by negotiating the security parameters. This includes agreeing on the encryption and authentication algorithms to be used, and authenticating the identity of the devices.

The second phase, known as the IKE Phase 2, uses the secure channel established in Phase 1 to establish the shared secret keys that will be used for encrypting and authenticating the data. It is during this phase that ESP and AH protocols can be implemented to protect the data.

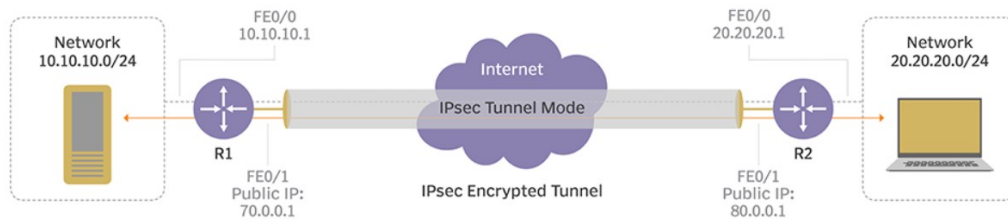


Figure 3: IPsec Tunnel Mode [5]

During the two phases of building an IPsec tunnel, ESP and AH protocols are used together or individually depending on the requirements of the organization. ESP provides encryption and confidentiality, while AH provides authentication and integrity. ESP encrypts the data and adds an ESP header to the packet, the AH on the other hand adds an Authentication header to the packet, which includes a digest of the packet's contents.

To sum up, Transport mode and Tunneling mode are two different ways of implementing IPsec, which are used to secure different types of IP traffic. Transport mode provides end-to-end security for individual IP packets, while Tunneling mode is used to create site-to-site VPNs by encrypting the entire IP packet. Building an IPsec Tunnel is a two-phase process, where IKE Phase 1 establishes a secure channel, and IKE Phase 2 establishes shared secret keys that will be used to encrypt and authenticate data.

2.4 Advantages and disadvantages

As shown on the previous points IPsec provides a range of features and security algorithms that make it a robust and secure solution for VPNs. However, like any technology, it also has its advantages and disadvantages.

One of the main advantages of IPsec is its flexibility. IPsec can be used in both transport mode and tunneling mode, making it suitable for a wide range of use cases such as E2EE, site-to-site VPNs, and inter-VLAN routing. It also supports a wide range of encryption and authentication algorithms, which provides organizations with the flexibility to choose the best algorithm to suit their security requirements.

IPsec also offers a high level of security by providing encryption, authentication, and key management. It uses IKE to establish a secure channel, and ESP and AH to provide encryption and authentication for the data. This ensures that the data transmitted over the IP network is protected from eavesdropping, tampering and replay attacks.

Another advantage is that IPsec is a standard protocol, which means it is widely supported by a wide range of devices and operating systems. This makes it easy to implement and manage in an organization's network.

On the other hand, one of the main disadvantages of IPsec is that it can add overhead to the network and can be processor-intensive. This can result in slower network performance and may require more resources and computational power on the devices implementing IPsec.

Another disadvantage is that IPsec can be complex to configure and manage, especially in large and complex networks. The protocol uses a variety of options and settings that can be difficult to navigate, and it requires careful configuration to ensure that it is secure and properly implemented.

In conclusion, IPsec is a widely used and robust protocol that offers a high level of security. However, it also has its advantages and disadvantages, such as flexibility, and high level of security, but also has the potential to consume processing power and requires careful configuration to ensure security. It's important for organizations to consider the trade-offs and to choose the solution that best meets their security requirements and network infrastructure.

3 WireGuard

WireGuard is a relatively new and open-source VPN protocol that aims to provide a simpler and more secure alternative to existing VPN protocols such as IPsec and OpenVPN. It was first introduced in 2015 and is based on a simplified security model that aims to provide faster performance and better security.

WireGuard is designed to be easy to configure, manage and audit, and it is also designed to be fast, lightweight and efficient. One of its core features is the usage of modern cryptographic primitives, Noise protocol for key exchange and the usage of kernel space, making it an interesting alternative to the older VPN protocols. WireGuard is becoming increasingly popular among network administrators and privacy enthusiasts alike, but it is not yet as widely adopted as other VPN protocols, and support for WireGuard is not yet as widespread.

3.1 Protocols and Features

WireGuard uses a unique key exchange mechanism, a technique called "no handshake" where the key is exchanged and authenticated before it is even used. This is different from other VPN protocols, such as IPsec and OpenVPN, which use a traditional handshake process to establish a secure connection. In a traditional handshake process, the two parties must first exchange a series of messages in order to negotiate the security parameters and establish a secure connection. This process can take a significant amount of time, especially in adverse conditions such as high packet loss or low bandwidth. WireGuard's "no handshake" approach eliminates the need for this traditional handshake process by pre-sharing the keys and performing the key exchange and authentication before the connection is even established. This allows WireGuard to establish a secure connection quickly and efficiently, even in adverse conditions.

The "no handshake" feature of WireGuard is made possible by the use of the Noise protocol for key exchange, which is a modern key exchange algorithm that is designed to be efficient and secure. In the Noise protocol, the two parties start with pre-shared static public keys and use them to establish a secure channel. The key exchange is performed using a single message, which includes the payloads for both parties, providing both the key and the proof of authenticity of the sender.

Another feature of WireGuard is that it can be implemented in the kernel, which provides better performance than other VPN protocols that are implemented in userspace. This allows WireGuard to handle a high number of connections and handle high-speed data transfer.

To reiterate, WireGuard is designed to be simple and efficient, while still providing a high level of security. It uses modern cryptographic algorithms and key exchange methods, which are considered more secure than older methods. WireGuard's implementation in kernel space and its unique key exchange mechanism make it performant, and its virtual interfaces and peers mechanism make it easy to manage. The built-in roaming support also allows for a seamless experience for users when switching between networks.

3.2 Security algorithms

WireGuard is built around the principle of simplicity and uses a minimal number of cryptographic primitives, making it more auditable and secure. The protocol uses state-of-the-art cryptographic algorithms to provide a high level of security, such as Curve25519 for key exchange, ChaCha20 for encryption and Poly1305 for authentication.

Curve25519 is a particular elliptic curve that can be used for Diffie-Hellman key exchange. It is designed to provide a high level of security and is considered to be one of the most secure elliptic curves available. The algorithm uses a base point (also known as generator) and the private key of each party to generate a public key, and then the two public keys are used to generate a shared secret key. This shared secret key can then be used to establish a secure connection. The advantage of using Curve25519 over other curves is that it is resistant to a number of known attacks and is considered to be one of the most secure elliptic curves available.

ChaCha20 is a stream cipher that is designed to be fast, secure and efficient. It is based on the Salsa20 algorithm, but uses a different set of round operations. The main difference between ChaCha20 and other stream ciphers is that it uses a unique 256-bit key and a 64-bit nonce. This nonce is used to ensure that each message encrypted with the same key generates a different ciphertext. This feature is crucial in order to prevent the reuse of a key and a nonce.

Poly1305 is an authenticator that is designed to provide data integrity and authentication in a very efficient way. It uses a 256-bit key and a 128-bit message to generate a 128-bit tag. It works by splitting the message into 16-byte blocks and then performs a series of mathematical operations on each block. The result is a unique tag that is associated with the message. This tag can then be used to verify the integrity and authenticity of the message at a later time. The poly1305 is considered to be a very efficient and secure authenticator, it is resistant to known attacks and it's widely used in combination with encryption algorithms such as ChaCha20.

In short, Curve25519 is an efficient and secure elliptic curve that is used for key exchange in WireGuard. It provides a high level of security by generating a shared secret key between two parties. ChaCha20 is a stream cipher that encrypts the data in WireGuard, it's designed to be fast and efficient, using a unique 256-bit key and a 64-bit nonce. The Poly1305 is an authenticator that provides data integrity and authentication. It uses a 256-bit key and a 128-bit message to generate a 128-bit tag. All these algorithms are carefully chosen and used in a specific way in the WireGuard to ensure a secure, efficient and hard to attack VPN connection.

3.3 Design and Implementation

WireGuard's design and implementation is built around the concept of "virtual interfaces" and "peers". A virtual interface is a virtual network interface that is created by the WireGuard software and used to establish a VPN connection. Each virtual interface is associated with a unique private key, and this key is used to encrypt and authenticate the data that is transmitted over the VPN connection. Peers are the remote parties that the virtual interfaces connect to. Each peer is identified by a unique public key, and this key is used to establish a secure connection.

WireGuard's design also allows for easy management of the VPN connections. Network administrators can easily establish, manage, and terminate VPN connections using the virtual interfaces and peers concept. WireGuard also has built-in support for roaming, which means it can automatically re-establish a VPN connection when a device switches between networks.

In terms of implementation, WireGuard is open-source and available for a wide range of platforms including Windows, Linux, and macOS, as well as for mobile platforms like Android and iOS. The source code is available for review and can be integrated to other systems, it is also easy to configure, even for non-technical users, with just a few simple commands.

In brief, WireGuard is designed to be simple, lightweight, efficient and easy to implement, manage and audit. The virtual interfaces and peers concept simplifies the configuration and management of VPN connections, and the built-in support for roaming allows for a seamless experience for users when switching between networks. The open-source nature of WireGuard makes it accessible and adaptable, it's available for multiple platforms, and it is easy to configure and integrate to other systems.

3.4 Advantages and disadvantages

WireGuard has several advantages over other VPN protocols, including:

- **Simpleness:** WireGuard is created with simplicity in mind, making it simple to implement, maintain, and audit. The protocol is more secure and less prone to errors thanks to the limited amount of cryptographic primitives it uses;
- **Efficiency:** WireGuard is designed to be lightweight and efficient, which makes it suitable for use on a wide range of devices, including mobile devices and low-powered devices;
- **Speed:** WireGuard establishes a secure connection quickly and effectively, even under challenging circumstances, thanks to its "no handshake" approach to key

exchange;

- **Security:** WireGuard uses a number of modern and secure algorithms, such as the Noise protocol, Curve25519, ChaCha20, and Poly1305, to provide a high level of security for its VPN connections;
- **Roaming support:** WireGuard has built-in support for roaming, which allows it to automatically re-establish a VPN connection when a device switches between networks;
- **Size:** One of its great advantages is its size. Wireguard clocks in at about 4,000 lines of code compared to 600,000 for OpenVPN or 400,000 for IPsec. Fewer lines mean the code is easier to inspect, and there are fewer places for bugs to pop up.

However, WireGuard also has some disadvantages to consider:

- **Lack of support for legacy systems:** WireGuard is a relatively new VPN protocol and may not be supported by older devices or systems;
- **Limited features:** WireGuard does not have as many features as some other VPN protocols, such as IPsec or OpenVPN, which may be a disadvantage for some users;
- **Limited scalability:** WireGuard is currently limited to a relatively small number of peers per interface, which may not be suitable for large-scale VPN deployments;
- **Limited device and platform support:** WireGuard may not be supported by some devices or platforms, or the support may be limited and not as well-maintained as other VPN protocols.

In summary, WireGuard is a fast, efficient, and secure VPN protocol that is designed to be simple and easy to implement. However, it is not yet widely supported by commercial VPN services, Network Devices and it lacks some features that other protocols provide. Additionally, its availability is currently limited to certain platforms. Its benefits and drawbacks should be taken into account when considering it as a solution for a VPN connection.

4 OpenVPN

OpenVPN is an open-source VPN solution that was first released in 2001. OpenVPN was developed as an alternative to proprietary VPN solutions that were avail-

able at the time. The development of OpenVPN was driven by the need for a more flexible and customizable VPN solution that would be suitable for a wide range of use cases, from personal use to large-scale enterprise deployments. Over the years, OpenVPN has evolved to include new features and capabilities, making it one of the most widely used open-source VPN solutions today.

This protocol offers a range of features to ensure secure and reliable connections. OpenVPN makes use of the OpenSSL library, providing a comprehensive set of security features such as encryption, authentication, and digital certificate management. It offers a lot of flexibility as it supports a variety of encryption algorithms and authentication methods, making it suitable for a wide range of use cases and adaptable to different security requirements. The protocol can be configured in both point-to-point and site-to-site configurations and can run on both TCP and UDP protocols. These factors make it accessible, adaptable and available for a wide range of platforms, such as Windows, Linux, macOS, Android and iOS.

4.1 Protocols and Features

OpenVPN is a widely used open-source solution for creating Virtual Private Networks (VPNs). This protocol offers a range of features to ensure secure and reliable connections. At the core of OpenVPN's operations is the OpenSSL library, which provides a wide range of cryptographic functions, including encryption, authentication, and digital certificate management. The protocol uses SSL/TLS for data encryption and authentication, which establishes a secure and encrypted connection between the client and the server. This provides a high level of security for the data transmitted over the VPN connection.

OpenVPN supports both TCP and UDP transport protocols, which means that it can adapt to different network conditions. For example, OpenVPN over UDP is usually faster than OpenVPN over TCP, but it can be less reliable in certain scenarios. The protocol also uses tunneling to establish a VPN connection, this means that it creates a virtual "tunnel" between the client and the server, through which all the traffic is routed. This technique allows the VPN connection to function as a separate and secure network connection, independent of the underlying public network.

Authentication is another important aspect of OpenVPN, the protocol supports various authentication methods including certificate-based and pre-shared key-based authentication, this way the server can be sure that the client is authorized to access the network. Additionally, OpenVPN supports compression, which helps to reduce the amount of data that needs to be transmitted over the VPN connection, which improves performance and reduces bandwidth usage. It also supports optimization

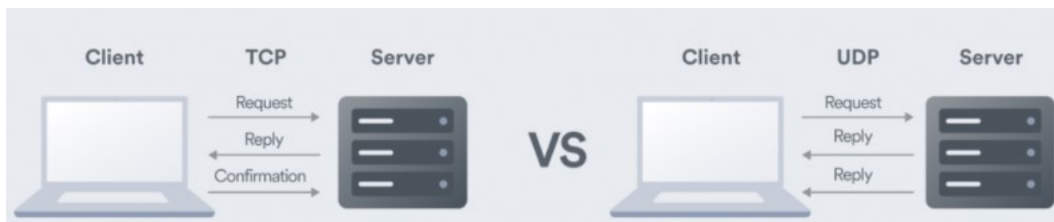


Figure 4: The two modes OpenVPN offers [1]

techniques such as pushing and pulling options to improve performance.

In brief, OpenVPN is a highly configurable and adaptable protocol that uses a range of protocols and features to establish secure and reliable VPN connections. It uses OpenSSL as the base of its operations, and it uses SSL/TLS for encryption and authentication. It supports both TCP and UDP, this provides flexibility and adaptability. Tunneling ensures a separate and secure connection over the underlying network, multiple authentication methods provide secure access, and it supports optimization techniques to improve performance. Finally, it's available and easily implementable on multiple platforms.

4.2 Security Algorithms

OpenVPN uses a number of security algorithms to ensure the security of the VPN connection. These algorithms are used for encryption, authentication, and digital certificate management.

Encryption algorithms: OpenVPN supports a wide range of encryption algorithms, such as AES, Blowfish and Camellia. These encryption algorithms are used to encrypt the data transmitted over the VPN connection, to ensure that the data is protected from unauthorized access or tampering. The encryption algorithm used can be configured to suit the security requirements of the network, for example, AES is widely used for its high level of security, whereas Blowfish is known for its fast encryption speed.

Authentication algorithms: OpenVPN supports several authentication algorithms, such as RSA(Rivest-Shamir-Adleman), a widely used public key encryption algorithm, DSA (Digital Signature Algorithm) which uses digital signature to provide authentication and ECDSA (Elliptic Curve Digital Signature Algorithm) which uses digital signature based on Elliptic Curve cryptography. These algorithms are used to authenticate the client and server, ensuring that only authorized clients are able to establish a VPN connection.

Digital certificate management: OpenVPN uses digital certificates to authenticate the server and the clients. It uses X.509 digital certificates, which are the

standard for public key infrastructure (PKI) and can be signed by a trusted certificate authority (CA) or can be self-signed. This provides a secure way to manage the identification of the clients and servers. Once the certificate is validated, the client and the server establish a secure connection using the encryption and authentication algorithms discussed earlier.

In short, OpenVPN uses a combination of security algorithms including encryption algorithms such as AES, Blowfish and Camellia, authentication algorithms such as RSA, DSA and ECDSA, and digital certificate management based on X.509 standard, to provide a secure and reliable VPN connection. These algorithms work together to encrypt the data transmitted, authenticate the clients and servers and provide secure identification and authentication of the clients and servers. This ensures that the data is protected from unauthorized access or tampering and that only authorized clients can establish a VPN connection.

4.3 SSL/TLS based VPN

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that are widely used to establish secure connections between clients and servers. SSL and TLS are responsible for establishing a secure and encrypted connection between the client and the server in OpenVPN.

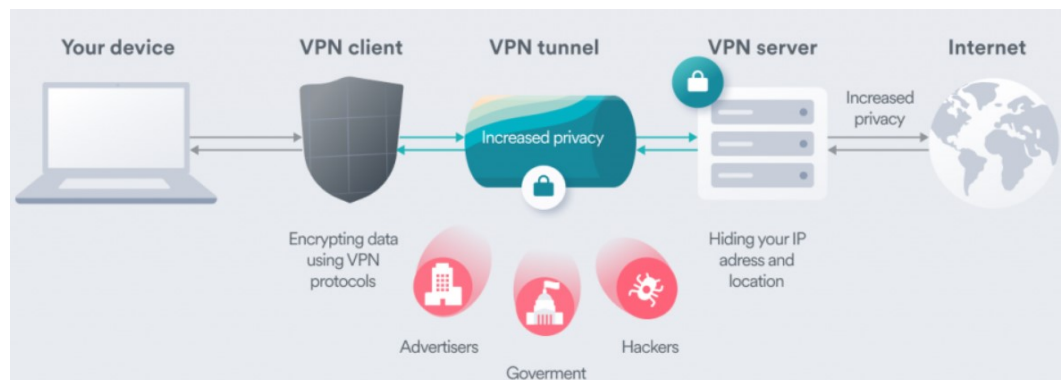


Figure 5: Connection between a client and a server using OpenVPN [1]

When an SSL/TLS connection is established, the client and the server initiate a "handshake" process. This process begins with the client sending a "ClientHello" message to the server, which contains information such as the SSL/TLS version being used, a list of supported cipher suites, and a randomly generated session ID. The server responds with a "ServerHello" message, which contains information such as the SSL/TLS version being used, the selected cipher suite, and the server's digital certificate.

The digital certificate contains the server's public key and is used to authenticate the server. It is signed by a trusted certificate authority (CA) and can be verified by the client using the CA's public key. Once the server's digital certificate is verified, the client generates a "pre-master secret", encrypts it using the server's public key and sends it to the server.

The server then uses its private key to decrypt the "pre-master secret" and both the client and server use the "pre-master secret" and a set of agreed upon algorithms to generate a "master secret". The "master secret" is then used to generate session keys, which are used to encrypt and decrypt the data transmitted over the connection.

Once the SSL/TLS connection is established, the client and server use the session keys to encrypt and decrypt the data transmitted over the connection. The encryption algorithm used is determined by the cipher suite selected during the handshake process.

To ensure the security of the connection, SSL and TLS also provide a mechanism for verifying the identity of the server, known as server authentication. This is achieved through the use of digital certificates, which are verified by the client using the CA's public key. Additionally, SSL and TLS use message authentication codes (MAC) to ensure the integrity of the data transmitted over the connection. The MAC is a cryptographic hash function that is applied to the data, along with a secret key, to generate a message authentication code. The MAC is then sent along with the data, and the receiving party verifies the integrity of the data by recomputing the MAC using the same secret key and comparing the results. If the computed MAC matches the one sent, the data has not been tampered with during transmission.

Another important aspect of SSL and TLS is the protection of the encryption keys from potential attackers. To achieve this, SSL and TLS use key exchange methods such as Diffie-Hellman, which allows the client and server to securely exchange encryption keys without them being intercepted.

In summary, SSL and TLS provide the foundation for a secure and encrypted connection in OpenVPN. The SSL/TLS handshaking process is used to authenticate the server, establish a secure connection, and exchange encryption keys. The connection is secured by the use of digital certificates, encryption algorithms, and message authentication codes (MACs) which ensures the integrity of the data and protects the encryption keys from potential attackers. With the help of SSL/TLS, OpenVPN creates an impenetrable layer of security for the data transmitted over the connection, providing a secure VPN connection.

4.4 Advantages and Disadvantages

OpenVPN offers a range of features to ensure secure and reliable connections, making it a popular choice among users. However, like any other technology, OpenVPN has its own set of advantages and disadvantages.

One of the main advantages of OpenVPN is its high level of security. The protocol uses SSL/TLS for data encryption and authentication, which establishes a secure and encrypted connection between the client and the server. Additionally, it supports a wide range of encryption algorithms, such as AES, Blowfish and Camellia, which can be configured to suit the security requirements of the network. It also supports various authentication methods including certificate-based and pre-shared key-based authentication, which ensures that only authorized clients are able to establish a VPN connection.

Another advantage of OpenVPN is its flexibility. It supports both TCP and UDP transport protocols, which means that it can adapt to different network conditions. The protocol also uses tunneling to establish a VPN connection, which creates a virtual "tunnel" between the client and the server, through which all the traffic is routed. This technique allows the VPN connection to function as a separate and secure network connection, independent of the underlying public network.

OpenVPN is also highly configurable, allowing for a wide range of options and settings to be configured to suit the needs of different environments and use cases. It also provides support for various operating systems, including Windows, Linux, macOS, and mobile platforms such as Android and iOS, making it accessible to a wide range of users.

On the other hand, one of the main disadvantages of OpenVPN is that it can be relatively complex to set up and configure. The wide range of options and settings available can make the setup and configuration process more challenging, particularly for users who are less technically savvy. Additionally, the SSL/TLS handshaking process, that is required to establish the connection, can introduce additional overhead which can affect the performance of the VPN connection. Also the use of digital certificates and key management can be complex, making it less suitable for novice users.

To reiterate, OpenVPN is a widely used open-source solution for creating VPNs. It offers a high level of security, flexibility, and configurability, making it a popular choice among users. However, it also has its own set of disadvantages, such as a complex setup and configuration process and a possible performance overhead, which can make it less suitable for novice users. Additionally, the use of digital certificates and key management can be complex for some users.

5 Theoretical comparison between IPsec, WireGuard, and OpenVPN

When it comes to comparing IPsec, WireGuard, and OpenVPN, there are several key factors to consider, including protocol, features, security algorithms, and performance.

In terms of protocol, IPsec is a mature and widely-used protocol, which has been around for over two decades. It uses a combination of the Internet Key Exchange protocol for key management and the Encapsulating Security Payload and Authentication Header protocols for encryption and authentication. WireGuard, on the other hand, is a relatively new protocol that has been designed with the goal of simplifying VPN connections, it uses the Noise protocol for encryption and key exchange and it focuses on a minimalistic design, aiming for higher performance and more secure connections. OpenVPN uses SSL/TLS for data encryption and authentication, and also supports both TCP and UDP transport protocols, it also uses OpenSSL library to provide a comprehensive set of cryptographic functions.

In terms of features, IPsec offers a wide range of options, including support for both transport mode and tunneling mode, which can be useful in different situations. WireGuard has a minimalist design, with fewer options compared to IPsec and OpenVPN, but still provides all the necessary features to establish a secure VPN connection. OpenVPN is highly configurable, allowing for a wide range of options and settings to be configured to suit the needs of different environments and use cases. Additionally, it also provides support for various operating systems and mobile platforms.

Security algorithms are another key consideration when comparing these three protocols. IPsec supports a wide range of encryption algorithms and authentication methods, such as AES, Blowfish, RSA and DSA among others. WireGuard uses the Noise protocol, which aims to provide more security than older methods, and also supports the use of ChaCha20 and Poly1305 for encryption and authentication. OpenVPN uses a number of security algorithms including encryption algorithms such as AES, Blowfish and Camellia, authentication algorithms such as RSA, DSA and ECDSA, and digital certificate management based on X.509 standard, to provide a secure and reliable VPN connection.





Finally, in terms of performance, WireGuard has been reported to have a high performance, due to its minimalistic design and the use of modern cryptographic algorithms, which have less overhead than some of the older algorithms used by IPsec and OpenVPN. Additionally, WireGuard has a relatively low latency, making it well-suited for real-time applications such as video conferencing or online gaming.

IPsec, on the other hand, has been reported to have higher latency and overhead due to the complexity of the protocol and the use of older algorithms. OpenVPN has also been reported to have higher latency than WireGuard, although not as high as IPsec, it can be affected by the SSL/TLS handshaking process and the use of digital certificates.

The properties of IPsec, WireGuard, and OpenVPN have been compiled in Table 1 for improved visibility and ease of comparison.

In conclusion, each of these protocols has its own set of strengths and weaknesses. IPsec is a mature and widely-used protocol that offers a wide range of options and good security. WireGuard is a new and simple protocol that focuses on high performance and security. OpenVPN is a configurable, flexible and comprehensive solution that offers good security and performance. The choice between the three depends on the specific requirements of the environment and the user's needs.

Table 1: Comparison between IPsec, WireGuard, and OpenVPN

	IPsec	WireGuard	OpenVPN
Released in	November 1998	September 2019	May 2001
OpenSource	Yes	Yes	Yes
Encryption algorithms	AES, Blowfish, and 3DES	Curve25519, ChaCha20 and Poly1305	AES, Blowfish and Camellia
Authentication Methods	ESP + AH	Noise protocol	RSA, DSA, ECDSA and X.509
Key Management	IKE	Noise protocol	SSL/TLS
Tunneling/Transport	Tunneling and Transport	Tunneling and Transport	Tunneling and Transport
Lines of Code	400.000	4.000	600.000
Latency	Higher than WireGuard and OpenVPN	Relatively Low	Higher than WireGuard but lower than IPsec
Overhead	Higher than WireGuard and OpenVPN	Relatively Low	Higher than WireGuard but lower than IPsec
Security	Very High	Very High	Very High
Ease of configuration and Management	Complex to configure and manage	Simple design and simple setup	Complex to configure and manage
Platform Support	  	    	    
Performance	Lower than WireGuard and OpenVPN	High	Lower than WireGuard but higher than IPsec

6 Practical Evaluation

Practical comparison is an integral aspect in determining the effectiveness and appropriateness of various VPN solutions. This section aims to provide an experimental setup and measurements of IPsec, WireGuard, and OpenVPN to understand their performance under different test scenarios. The outcomes of the tests, as well as a comprehensive analysis of the performance of each VPN solution, will be presented. A demonstration of the setup and configuration of each VPN solution will also be provided. The aim of this section is to gain a thorough understanding of the performance of each solution, and to identify the best VPN solution for various use cases and environments.

6.1 Experimental Setup and Initial Configurations

In order to evaluate the performance of different VPN solutions, virtual machines running Kali Linux will be set up on VirtualBox. The use of Kali Linux allows for easy access to VPN clients for IPsec, OpenVPN, and Wireguard, as well as tools such as iperf for measuring throughput. These virtual machines will be configured to establish VPN connections and transfer data of varying sizes to measure the impact of these solutions on connection speed. By comparing the results of these tests, the best VPN solution for each specific use case can be determined. All the steps are carefully explained in order to allow readers to replicate the process.

6.1.1 Methodology and Test Design

To perform the end-to-end encryption performance evaluation the following steps will be taken:

1. Install the required software and tools for each solution:
 - a. IPsec: Install an IPsec implementation such as StrongSwan.
 - b. WireGuard: Install the WireGuard software.
 - c. OpenVPN: Install the OpenVPN software.
2. Set up the end-to-end encryption:
 - a. IPsec: Configure encryption settings, including encryption algorithm, key exchange algorithm, and authentication method.
 - b. WireGuard: Generate encryption keys and configure encryption settings.
 - c. OpenVPN: Generate encryption keys and configure encryption settings, including encryption algorithm and key exchange algorithm.

3. Test the end-to-end encryption using iperf.
4. Repeat for different encryption algorithms and configurations to gather data for the comparison.

In the case of a site-to-site VPN the situation is a bit more complex. Now, four virtual machines will be used. The steps are as follows:

1. VPN Connection Establishment: A site-to-site VPN connection is established between the two gateways (first two virtual machines) using each VPN solution. The connection is configured with different encryption algorithms and configurations to gather data for the comparison.
2. Traffic Generation: Traffic is generated between two nodes (other two virtual machines, for example, the ones used in end-to-end) using iperf to measure the performance of each VPN solution in terms of throughput.
3. Again, repeat for different encryption algorithms and configurations to gather data for the comparison.

The different tests that will be made for end-to-end and site-to-site are as follows:

1. No encryption (Benchmark test)
2. AES-256 with HMAC-SHA256 authentication in IPsec
3. 3DES with HMAC-SHA1 authentication in IPsec
4. Chacha20 with Poly1305 authentication in WireGuard
5. AES-256 with SHA-256 authentication in OpenVPN
6. Blowfish with SHA-384 authentication in OpenVPN

The tests include generating traffic of 10 KB, 100KB, 1MB, 10MB and 100MB and measuring the average performance for which amount.

6.1.2 Kali Linux and Virtual Box

Kali Linux is a widely-used open-source operating system for penetration testing and security auditing. It is based on Debian Linux and is specifically designed for ethical hackers, security researchers, and penetration testers. Kali Linux provides a wide range of tools and features for conducting various types of security testing, including network penetration testing, web application testing, wireless testing, and

more. It also offers a user-friendly interface and a variety of customization options for users to adapt the system to their specific needs.

In the context of this report, Kali Linux will be used as the operating system for four virtual machines created on VirtualBox. These virtual machines will be configured to test the performance of different VPN solutions in both end-to-end and site-to-site configurations.

Installing Kali Linux on VirtualBox is a straightforward process, but it requires a few steps to be followed.

First, you need to download the Kali Linux ISO file from the official website. Once you have the ISO file, you need to create a new virtual machine on VirtualBox. To do this, open VirtualBox, click on the "New" button, and enter a name for the virtual machine. Select "Linux" as the type and "Debian" as the version.

Next, you need to allocate resources to the virtual machine. You can adjust the amount of RAM and the number of virtual processors according to your host system's resources.

After setting up the basic configuration, you need to create a virtual hard drive for the virtual machine. You can choose between a virtual hard disk file or a physical hard disk. Once you have created the virtual hard drive, you can proceed with the installation process.

To install Kali Linux, click on the "Start" button, and select the Kali Linux ISO file that you downloaded earlier. The virtual machine will boot from the ISO file, and you will be prompted to select your language and keyboard layout.

After that, you will need to go through the installation process, which includes partitioning the hard drive, configuring the network, and setting a password for the root user. Once the installation is completed, you should be able to boot into the newly installed Kali Linux virtual machine.

Additionally, it is necessary that the virtual machines all run in a Bridged Network. A Bridged Network is a network setup where the virtual machines are connected to the host machine's network adapter. This means that the virtual machines will be able to communicate with other devices on the same network as the host machine, including other virtual machines and physical devices. This is why it is necessary to configure the virtual machines to operate in a bridged network when using VirtualBox. By doing this, you are essentially giving the virtual machine its own IP address on the network and allowing it to function as an independent device. To set up a Bridged Network in VirtualBox, you can go to the Network settings of the virtual machine and select "Bridged Adapter" from the drop-down menu for the "Attached to" option. Then, select the host machine's network adapter from the list of available adapters and click OK. The virtual machine should now be connected to

the network and able to communicate with other devices. Figure 6 has a simplified demonstration of what is a Bridged Network.

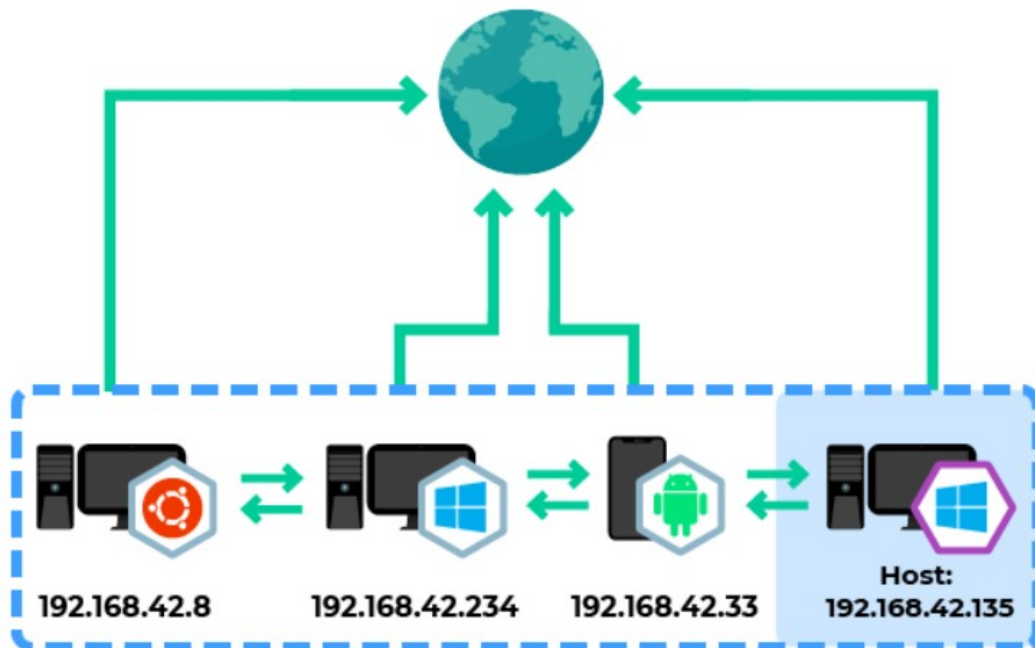


Figure 6: Bridged network diagram [25]

It's important to note that, while Kali Linux is a powerful tool for penetration testing and security research, it should be used responsibly and in compliance with all applicable laws and regulations. It's also important to be aware of the fact that, as it's a tool for security professionals and researchers, it's not intended for general users, and it may have stability issues or compatibility problems with some hardware.

6.1.3 Installing the Softwares

First, it is necessary to ensure that the Kali Linux virtual machine is up-to-date by running the commands:

```
sudo apt-get update
sudo apt-get upgrade
```

First, the necessary packages for IPsec can be installed by running the command:

```
sudo apt-get install strongswan
```

To install WireGuard run the following command:

```
sudo apt-get install wireguard
```

Once the installation is complete, the WireGuard tools will be available in the system. To check if the installation was successful, you can run the following command:

```
wg
```

It should show the help message for the command.

The OpenVPN package can be installed by running the command:

```
sudo apt-get install openvpn
```

Now that the VPN solutions have all been installed we can also install the iperf tool which will be used to perform the throughput tests. This can be done by using the command:

```
sudo apt-get install iperf
```

For our tests to work it's first necessary to enable packet forwarding on the virtual machine. This is done by modifying the `sysctl` configuration file. To do this, we open the file using the nano editor: `sudo nano /etc/sysctl.conf` and then look for the following lines and uncomment them.

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
```

After making these changes, we load the new settings by executing `sudo sysctl -p`.

Since we'll be using four virtual machines in our tests we can now shut down the virtual machine and create three full clones of it with different addresses. This process will take a while.

6.2 E2EE Tests

As explained previously, the objective in this end-to-end encryption tests is to configure two virtual machines as endpoints and have a secure channel between them, allowing secure traffic to be generated. Figure 7 demonstrates the scheme of the test.

6.2.1 No Encryption (Benchmark)

In this first section, we are conducting a benchmark test for the end-to-end encryption between two virtual machines (VM1 and VM2) to measure the throughput of the connection. The purpose of this test is to establish a baseline of performance before implementing encryption to the data transfer.

On VM1, start the iperf server by executing the following command:



Figure 7: E2EE test scheme

```
iperf -s
```

On VM2, start the iperf client by executing the following command, replacing the IP address of VM1, that can be discovered by running `ifconfig` and the data size with one of 10 KB, 100KB, 1MB, 10MB and 100MB:

```
iperf -c [IP address of VM1] -n [data size in bytes]
```

Repeat this command several times to get an averaged value for which of the data sizes and take note of the values of the time and throughput.

The objective from now on is to repeat this process but with a secure channel provided by each of the solutions that will be tested. By taking note of the values of the throughput it is possible to compare all the solutions.

6.2.2 AES-256 with HMAC-SHA256 authentication in IPsec

First on the list is IPsec with AES-256 and HMAC-SHA256 authentication. After installing the **Strongswan** packages a file called `ipsec.conf` is created in the `/etc` folder. This file is used to configure the connection between the two nodes, that is, VM1 and VM2. Open the file on both machines using the command `sudo nano /etc/ipsec.conf` and copy the following instructions:

```
config setup
    charondebug="all"
    uniqueids=yes

conn newconn
    type=transport
    auto=start
    keyexchange=ikev2
    authby=secret
    left=[IP address of VM1]
    leftsubnet=[Subnet of VM1]
    right=[IP address of VM2]
```

```
rightsubnet=[Subnet of VM2]
ike=aes256-sha256-modp1024!
esp=aes256-sha256!
aggressive=no
keyingtries=%forever
ikelifetime=28800s
lifetime=3600s
dpddelay=30s
dpdtimeout=120s
dpdaction=restart
```

Now that the connection is configured it's time to start it on both machines, using the command:

```
sudo ipsec start
```

We can also check if the connection was successful by using the command:

```
sudo ipsec status
```

All that's left to do is run the `iperf` tests, take note of the results and close the session in the end:

```
sudo ipsec stop
```

6.2.3 3DES with HMAC-SHA1 authentication in IPsec

For the sake of good comparisons it's essential to run the tests multiple times with different algorithms in order to spot differences in performance. Now that the `ipsec.conf` file is configured, all that is needed to do to try different algorithms is change two lines of the file:

```
ike=3des-sha1-modp1024!
esp=3des-sha1!
```

Rerun the `iperf` tests, take note of the results and close the session in the end.

6.2.4 Chacha20 with Poly1305 authentication in WireGuard

In order to test the end-to-end encryption of WireGuard we will configure the two virtual machines (VM1 and VM2) to act as peers, A and B respectively. This involves exchanging public keys between the peers and performing a minimal amount of configuration. The first order of business is to create a new configuration file at `/etc/wireguard/wg0.conf` and set its permissions to be restricted:

```
(umask 077 && printf "[Interface]\nPrivateKey = " | sudo tee
/etc/wireguard/wg0.conf > /dev/null)
```

```
wg genkey | sudo tee -a /etc/wireguard/wg0.conf | wg pubkey |
sudo tee /etc/wireguard/publickey
```

The second command generates a private key using the `wg` command and writes it directly to the `/etc/wireguard/wg0.conf` file. The private key is then piped back into the `wg pubkey` command to derive the associated public key, which is saved in a file located at `/etc/wireguard/publickey`. This file contains the public key that will be exchanged with the second server to complete the configuration.

Next, we will open the two configuration files created and edit them:

```
sudo nano /etc/wireguard/wg0.conf
```

The files are divided into an Interface area and a Peer area. It should contain the public and private keys of each peer, as well as the addresses and ports that they will use to communicate with each other. The file for VM1 is:

```
[Interface]
PrivateKey = generated_private_key
ListenPort = 5555
SaveConfig = true
Address = [IP address of VM1]/24

[Peer]
PublicKey = [Public key of VM2]
AllowedIPs = [IP address of VM2]/32
Endpoint = [IP address of VM2]:5555
```

Very similarly, the file for the second VM is:

```
[Interface]
PrivateKey = generated_private_key
ListenPort = 5555
SaveConfig = true
Address = [IP address of VM2]/24

[Peer]
PublicKey = [Public key of VM1]
AllowedIPs = [IP address of VM1]/32
Endpoint = [IP address of VM1]:5555
```

Now we can start the WireGuard service on both VMs by using the command:

```
sudo systemctl start wg-quick@wg0
```

Additionally, we can check the connection by using the command `sudo wg`. This will provide details about the connection. Now that the two peers are successfully connected using WireGuard we can conduct the `iperf` tests and take note of the results obtained.

6.2.5 AES-256 with SHA-256 authentication in OpenVPN

The configuration of OpenVPN is much more complex than what it was for the other two solutions. It involves more steps and more configurations. All the steps necessary to have a secure channel between VM1 and VM2 are stated next. The first Virtual Machine will act as the OpenVPN server and the second as the OpenVPN client, so every time server and client are mentioned in the following OpenVPN sections we are talking about the two virtual machines.

1. Create an OpenVPN Configuration Directory: On both the client and server machines, create a directory to store your OpenVPN configuration files.

```
sudo mkdir /etc/openvpn/server  
sudo mkdir /etc/openvpn/client
```

2. Generate the CA (Certificate Authority) Files: On the server machine, run the following command to generate the CA files. Replace the values for `KEY_COUNTRY`, `KEY_PROVINCE`, `KEY_CITY`, `KEY_ORG`, and `KEY_EMAIL` with your desired values. These variables are fields that represent various parts of your organizational identity, and are used to create the SSL/TLS certificate when setting up an OpenVPN server.

```
sudo openssl req -nodes -new -x509 -keyout ca.key -out ca.crt -subj  
"/C=KEY_COUNTRY/ST=KEY_PROVINCE/L=KEY_CITY/O=KEY_ORG/CN=KEY_EMAIL"
```

3. Generate the Server Key: On the server machine, run the following command to generate the server key.

```
sudo openssl genpkey -algorithm RSA -out server.key
```

4. Generate the Server Certificate Request: On the server machine, run the following command to generate the server certificate request. Replace the values for KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL with your desired values.

```
sudo openssl req -key server.key -new -out server.csr -subj  
"/C=KEY_COUNTRY/ST=KEY_PROVINCE/L=KEY_CITY/O=KEY_ORG/CN=KEY_EMAIL"
```

5. Sign the Server Certificate Request: On the server machine, run the following command to sign the server certificate request.

```
sudo openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -out  
server.crt
```

6. Generate the Client Key: On the client machine, run the following command to generate the client key.

```
sudo openssl genpkey -algorithm RSA -out client.key
```

7. Generate the Client Certificate Request: On the client machine, run the following command to generate the client certificate request. Replace the values for KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL with your desired values.

```
sudo openssl req -key client.key -new -out client.csr -subj  
"/C=KEY_COUNTRY/ST=KEY_PROVINCE/L=KEY_CITY/O=KEY_ORG/CN=KEY_EMAIL"
```

8. Sign the Client Certificate Request: On the server machine, run the following command to sign the client certificate request.

```
sudo openssl x509 -req -in client.csr -CA ca.cr
```

9. Configure the OpenVPN server: On the server machine, create a new OpenVPN configuration file by running the following command:

```
sudo nano /etc/openvpn/server.conf
```


In the `server.conf` file, copy and paste the following contents, making sure to replace the variables with your specific information:

```
port 1194
proto udp
dev tun
ca /path/to/ca.crt
cert /path/to/server.crt
key /path/to/server.key
dh /path/to/dh.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
keepalive 10 120
tls-auth /path/to/ta.key 0
key-direction 0
cipher AES-256-CBC
auth SHA256
comp-lzo
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
verb 3
```

Replace `"/path/to/ca.crt"`, `"/path/to/server.crt"`, `"/path/to/server.key"`, `"/path/to/dh.pem"`, and `"/path/to/ta.key"` with the actual paths to the certificate files on your server machine. Save and close the `server.conf` file.

10. Create an OpenVPN client configuration file.

```
sudo nano /etc/openvpn/client.conf
```

11. Add the following lines to the `client.conf` file:

```
client
dev tun
proto tcp
remote [IP address of server]
resolv-retry infinite
nobind
cipher AES-256-CBC
auth SHA256
user nobody
group nogroup
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
comp-lzo
verb 3
```

12. Start the OpenVPN server:

```
sudo systemctl start openvpn@server
```

13. Start the OpenVPN client:

```
sudo systemctl start openvpn@client
```

14. Check the status of the OpenVPN connection to ensure that it's up and running:

```
sudo systemctl status openvpn@server
sudo systemctl status openvpn@client
```

You should see a message indicating that the OpenVPN connection is active and the process is running. You now have a secure end-to-end encrypted connection between the two virtual machines using OpenVPN.

15. Finally, perform the `iperf` tests and take note of the results.

6.2.6 Blowfish with SHA-384 authentication in OpenVPN

In order to change the encryption and authentication algorithms used in OpenVPN, you need to modify the server and client configuration files. The server configuration file and the client configuration file both contain options that specify the encryption and authentication algorithms used. By default, OpenVPN uses AES-256 encryption and SHA-256 authentication, but these algorithms can be changed to any of the supported encryption and authentication algorithms.

To change the algorithms to Blowfish and SHA-384, you will need to modify the encryption and authentication options in both the server and client configuration files. First you need to locate the line that specifies the encryption algorithm and change it from AES-256-CBC to BF-CBC. Next, locate the line that specifies the authentication algorithm and change it from SHA256 to SHA384.

Once you have made the changes, save the files and restart the OpenVPN service on both the server and the client. This can be done by running the following command:

```
sudo systemctl restart openvpn@server  
sudo systemctl restart openvpn@client
```

After the service has restarted, the encryption and authentication algorithms will be changed to Blowfish and SHA-384, respectively and the `iperf` tests can be performed again.

6.2.7 Results of the Evaluation

The results of all the E2EE testing performed are represented in figures 8 to 12.

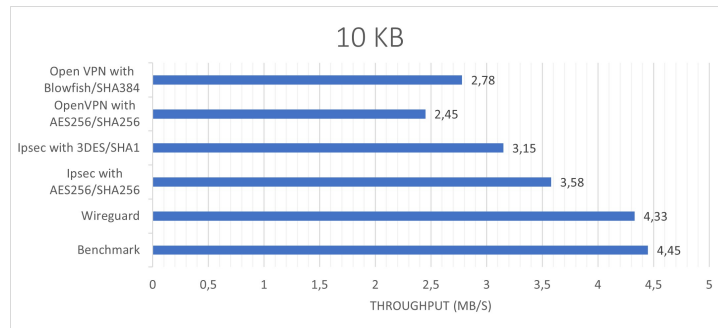


Figure 8: E2EE tests with 10 KB traffic

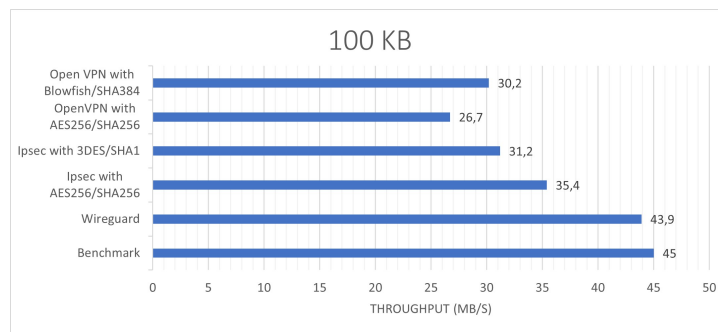


Figure 9: E2EE tests with 100 KB traffic

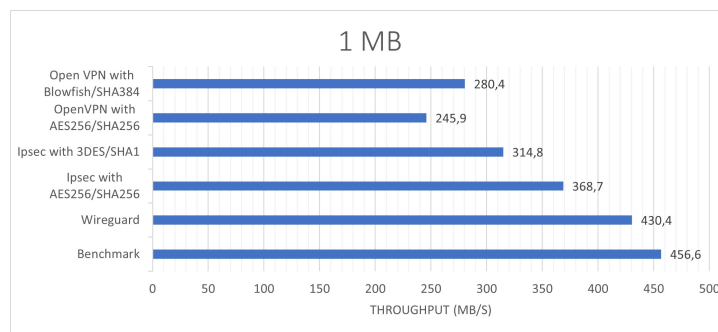


Figure 10: E2EE tests with 1 MB traffic

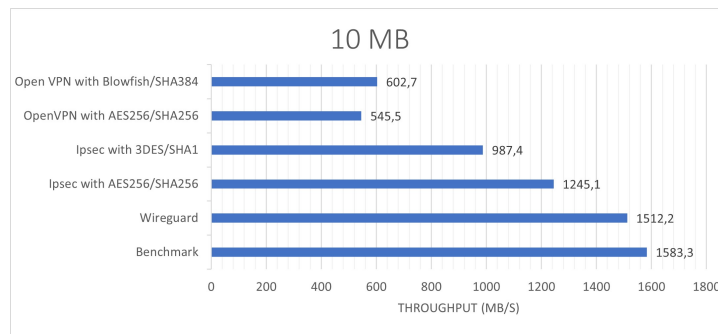


Figure 11: E2EE tests with 10 MB traffic

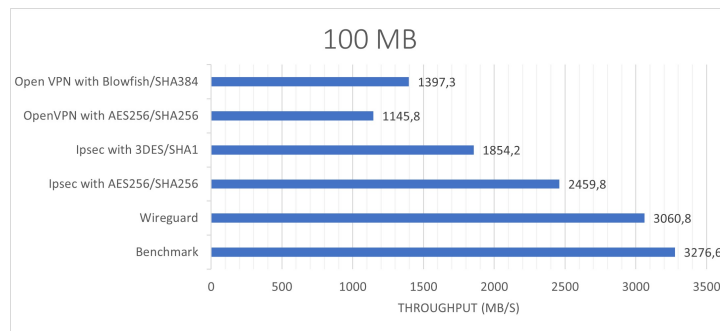


Figure 12: E2EE tests with 100 MB traffic

The results of the tests allow for multiple considerations. In the results of the tests performed, WireGuard was found to be extremely fast and efficient, with performance almost matching that of the benchmark test.

After Wireguard, the solution with the best performance is IPsec with AES 256 and then IPsec with 3DES. These results are expected as IPsec is more complex than Wireguard so it's normal that its results are slightly slower than Wireguard's. Also we can observe that AES is faster than 3DES. This slower performance could be due to the use of a weaker encryption algorithm.

OpenVPN, on the other hand, was significantly slower compared to both IPsec and WireGuard. Although OpenVPN with Blowfish encryption was faster than OpenVPN with AES-256 encryption, it was still slower than both IPsec configurations. AES-256 is slower due to its more complex encryption, but it will be more secure. Blowfish, on the other hand, is faster but less secure than AES-256. It is important to note that speed should not be the only factor to consider when choosing a VPN protocol.

The security of the VPN protocol is also of paramount importance. In this regard, for example, 3DES with SHA-1 encryption, although faster than OpenVPN with AES-256 encryption, is an outdated configuration that does not provide suffi-

cient security for current needs. As a result, OpenVPN with AES-256 encryption, despite its slower performance, would be a smarter choice over that specific IPsec configuration. The choice between different configurations ultimately depends on the desired balance between security and performance.

6.3 Site-to-site VPN Tests

As explained previously, the objective in this site-to-site VPN tests is to configure two virtual machines as gateways and two virtual machines as endpoints. The goal is to establish a secure and encrypted connection between the two endpoints through the gateways, so that data can be transmitted between the endpoints. Figure 13 demonstrates the scheme of the test.

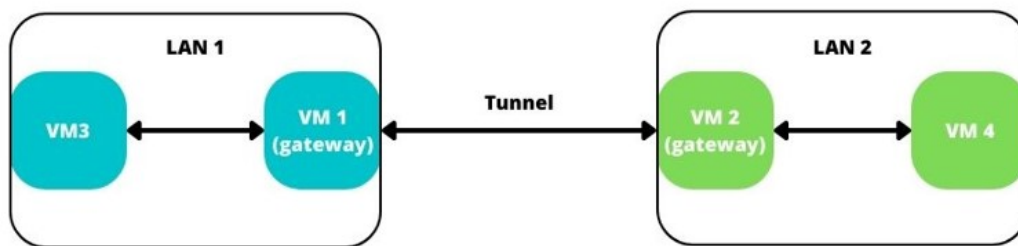


Figure 13: Site-to-Site VPN test scheme

VM1 and VM2 in this scenario will be the virtual machines used for the end to end tests, as they already have all the configurations, files, keys etc. created. This way it's only necessary to change a few things instead of starting from scratch. VM3 and VM4 don't need any special configurations in regard to the VPN solutions as it is the gateways job to handle the encryption/decryption tasks. Their job is simply to generate the traffic.

6.3.1 No encryption (Benchmark test)

First, we are conducting a benchmark test for the end-to-end encryption between the four two virtual machines to measure the throughput of the connection. The purpose of this test is to establish a baseline of performance before implementing encryption to the data transfer.

A simple way to do this is by confuring the routes of the traffic from VM3 to VM4. To do this use the commands: On VM3:

```
ip route add [IP address of VM4] via [IP address of VM1]
```

On VM1:

```
ip route add [IP address of VM4] via [IP address of VM2]
```

Now, when traffic is generated from VM3 to VM4, it will pass through all the machines. Run the `iperf` tests and take note of the results.

To reset the routes use the command:

```
sudo route flush
```

6.3.2 AES-256 with HMAC-SHA256 authentication in IPsec

To start off the configuration of the site-to-site VPN we first open the `ipsec.conf` files that we already have from the E2EE tests. This file has that `type = transport`. The first thing we need to do is change it to `type = tunnel` and then change the algorithms back to AES-256 with HMAC-SHA256 authentication. The rest of the configuration file is the same.

Next it is necessary to reconfigure the routes of the traffic for VM3 and VM4. The traffic generated should not go directly between them but pass through the gateways. In order to do this, type the following command on VM3:

```
ip route add [IP address of VM4] via [IP address of VM1]
```

And on VM4:

```
ip route add [IP address of VM3] via [IP address of VM2]
```

Now that the connections are configured it's time to start it on both machines (VM1 and VM2), using the command:

```
sudo ipsec start
```

All that's left to do is run the `iperf` tests between VM3 and VM4, take note of the results and close the session in the end.

6.3.3 3DES with HMAC-SHA1 authentication in IPsec

In order to change the algorithms we follow the same steps as for the E2EE tests. All that is needed is to open the `ipsec.conf` files of the gateway machines and change two lines of the file:

```
ike=3des-sha1-modp1024!  
esp=3des-sha1!
```

6.3.4 Chacha20 with Poly1305 authentication in WireGuard

In a Site-to-Site VPN setup using Wireguard, the gateways are responsible for creating a secure tunnel between the endpoints and ensuring that the data transmitted between them is properly encrypted and authenticated.

The situation here is similar to IPsec. Since we have already set up the configuration files and private and public keys during the E2EE tests, and the IP routing table during the previous step, all that is needed to be done now is to start the WireGuard service between the two gateways (VM1 and VM2). With the previous setup in place, we can simply start the WireGuard service on both gateways and initiate transmission between the endpoints (VM3 and VM4) and run the `iperf` tests.

6.3.5 AES-256 with SHA-256 authentication in OpenVPN

When it comes to Site-to-Site VPN using OpenVPN the situation is, again, the same. The config files are already setup and certificates are still acceptable so we can just start the connection between VM1 and VM2 and perform the tests. The routing table knows how to forward the traffic that will be generated by VM3 to VM4 so we can run the tests.

6.3.6 Blowfish with SHA-384 authentication in OpenVPN

Once more, the vital actions to change the algorithms for encryption and authentication were already explained. The only thing to do is to locate the line that specifies the encryption algorithm and change it to BF-CBC and the line that specifies the authentication algorithm to SHA384 and run the tests.

6.3.7 Results of the Evaluation

The results of all the E2EE testing performed are represented in figures 14 to 18.

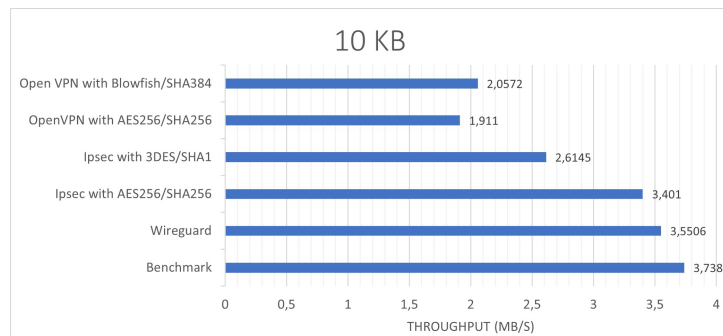


Figure 14: Site-to-Site tests with 10 KB traffic

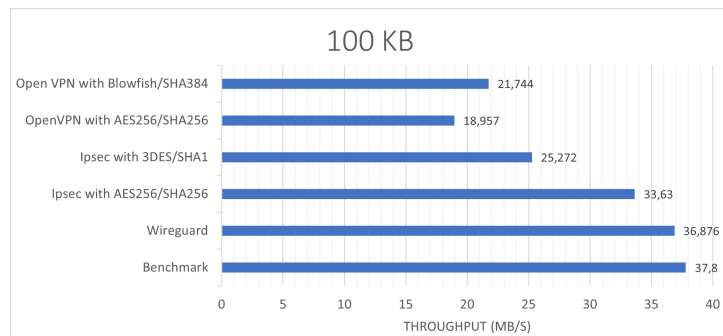


Figure 15: Site-to-Site tests with 100 KB traffic

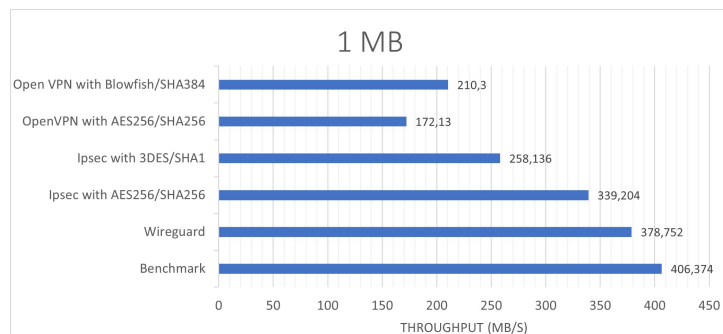


Figure 16: Site-to-Site tests with 1 MB traffic

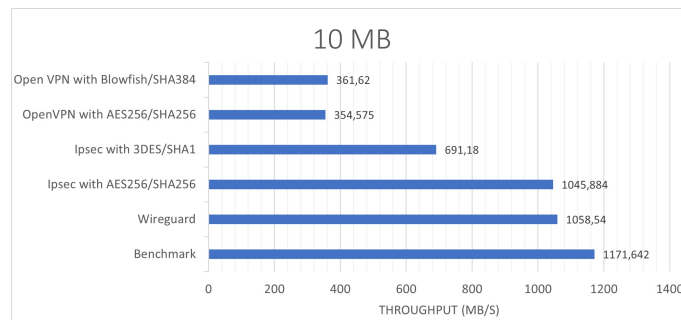


Figure 17: Site-to-Site tests with 10 MB traffic

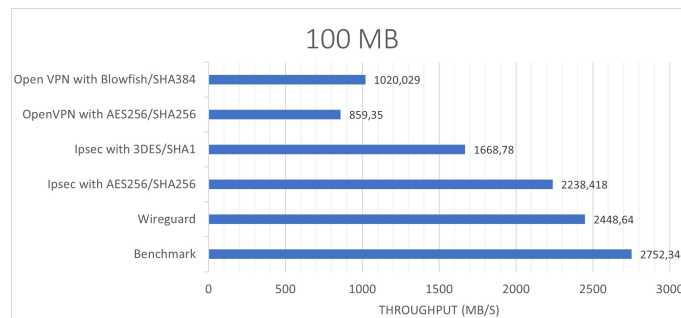


Figure 18: Site-to-Site tests with 100 MB traffic

Based on the results of the site to site VPN tests, it can be concluded that Wireguard is still the fastest solution among the three options. However, this time around, IPsec performed much closer to Wireguard in terms of throughput. Unlike the previous tests, where Wireguard significantly outperformed IPsec, the results this time were much closer and competitive. On the other hand, OpenVPN performed much worse compared to the other two solutions.

These results demonstrate that the choice of VPN solution for site to site connections is not just about raw speed, but also about compatibility with different systems, security features, and scalability. Wireguard's simple design, strong security features, and performance make it a good choice for many use cases, especially for smaller networks. IPsec, being a more established and widely used VPN solution, has the advantage of being compatible with a wider range of systems and having a more robust feature set. However, its performance is not as good as Wireguard's, so it might not be the best choice for latency-sensitive applications.

These results also showed that, compared to the end-to-end encryption tests, all solutions had a slower throughput. This indicates that the added complexity and additional routing channels involved in site-to-site VPNs can have a significant impact on performance. This is also due to the added overhead of encrypting and decrypting data at each hop.

7 Conclusions

7.1 Summary of the findings and results of the comparative evaluation

This report aimed to provide a comprehensive comparison of three widely used solutions for end-to-end and site-to-site VPN: IPsec, Wireguard, and OpenVPN. The comparison was done both theoretically and practically, covering the protocol, features, algorithms, and experimental setup and measurements.

Starting with the theoretical aspect, it detailed the underlying protocols and features of each VPN solution. IPsec, for example, uses IKE and ESP or AH protocols for key exchange and data protection, respectively, with the possibility of using different cryptographic algorithms such as AES or 3DES. Wireguard, on the other hand, uses the noise protocol for key exchange and encrypts data with ChaCha20 and authenticates with Poly1305. OpenVPN uses SSL/TLS protocols and can use various encryption algorithms such as AES or Blowfish.

On the practical aspect, a laboratory environment was created using virtualization software, specifically Oracle VirtualBox, to evaluate the performance of each VPN solution under the same conditions. Throughput was used as the main metric to evaluate the performance, and it was observed that different configurations and scenarios can lead to different results.

When considering end-to-end encryption, IPsec showed to have a good balance between security and performance, with Wireguard showing better performance but with a lower level of security. OpenVPN performed poorly on the throughput tests but, like IPsec, it also has a good balance between security and performance. For site-to-site VPN, both IPsec and Wireguard showed similar results, with OpenVPN showing slightly lower performance again.

Overall, it was observed that each VPN solution has its own strengths and weaknesses, and the best solution depends on the specific requirements of the network, such as performance, security, and compatibility. It should be taken into account that these results may vary depending on the specific configurations, devices, and scenarios.

This report and the accompanying demonstration aim to provide a comprehensive understanding of the VPN solutions, allowing a better decision to be made when choosing a solution for a specific network. This research can be used as a guide to understand the strengths and weaknesses of each solution and as a basis for further experimentation and research on VPN solutions.

7.2 Recommendation for the most suitable solution based on specific requirements and use cases

When choosing a VPN solution, it is important to consider specific requirements and use cases. Different solutions may have different strengths and weaknesses depending on the intended use. For example, end-to-end encryption (E2EE) is ideal for situations where privacy and security are top priorities, such as in financial transactions or communication of sensitive information. However, E2EE can come at a cost of reduced speed and may not be suitable for high-bandwidth applications. On the other hand, a site-to-site VPN may be better for connecting remote offices or for linking networks for resource sharing, as it provides a direct connection between the networks without the need for individual client software installations.

In a scenario where a small business needs to connect its remote employees to the main office network, a site-to-site VPN may be the best choice as it allows for easy and secure connection to the network resources.

For example in a hospital setting where sensitive patient information is being transmitted, E2EE may be the more suitable solution. The encrypted transmission of data ensures that even if the data is intercepted, it will be unreadable without the encryption key. This ensures the privacy and security of the patient information.

In a gaming scenario where speed and low latency are important factors, a VPN solution such as WireGuard may be the best choice. WireGuard has shown to have similar results to the benchmark test, meaning it is fast and suitable for high-bandwidth applications. This makes it a great choice for online gaming where a fast and reliable connection is essential.

In a large enterprise environment with strict security requirements, IPsec can be a better solution. IPsec provides a more secure encryption protocol compared to OpenVPN and it has been around for a long time, making it well-established and widely used. IPsec is also compatible with most firewalls, routers, and network devices, making it easier to implement and manage. Additionally, IPsec is a better choice for sites that need to be connected over a wide area network (WAN) as it provides high-speed performance. In this scenario, security and performance are the key considerations, and IPsec offers a more robust solution than OpenVPN.

In an environment where the client devices have limited resources and processing power, OpenVPN can be a better choice. OpenVPN can run on low-end devices such as smartphones, laptops, and tablets. In a scenario where remote workers need to connect to the company network, OpenVPN can provide a secure and efficient solution. It has a lightweight protocol that can run on most devices and its simplicity in setup and configuration makes it easy to deploy on a large scale. In this scenario,

the speed of the VPN may not be as critical as the ease of use and compatibility with a wide range of devices.

Ultimately, the choice of VPN solution will depend on specific requirements and use cases. It is important to weigh the trade-offs between speed, security, and ease of use when making a decision. When in doubt, seeking the advice of a network professional can help ensure that the most suitable solution is chosen for each specific need.

7.3 Discussion of the future trends and directions for end-to-end encryption and site-to-site VPN solutions

The future of end-to-end encryption (E2EE) and site-to-site VPN (Virtual Private Network) solutions is expected to be shaped by the increasing demand for secure and private communication over the internet. As the use of digital devices and services continues to grow, the need for secure communication will become increasingly important. In response to this demand, E2EE and Site-to-Site VPN solutions are likely to continue to evolve and improve, offering better performance, security, and ease of use.

In the case of E2EE, the trend is towards making this type of encryption more widely available and accessible, especially for mobile devices and cloud-based services. This will likely involve the development of new encryption algorithms and technologies, as well as the integration of E2EE into popular operating systems, applications, and services. The goal will be to provide users with a simple and secure way to communicate and share information, regardless of the device or platform they are using.

In the case of Site-to-Site VPN, the trend is towards offering more flexible and scalable solutions that can meet the needs of large and complex networks. This will likely involve the development of new VPN protocols and technologies that can support high-speed communication and large amounts of data, as well as the integration of Site-to-Site VPN with cloud-based services and other network infrastructure. The goal will be to provide organizations with a secure and reliable way to connect and communicate between their remote locations and the main corporate network.

In conclusion, the future of E2EE and Site-to-Site VPN solutions will be defined by the growing demand for secure and private communication over the internet. Both types of solutions are likely to continue to evolve and improve, offering better performance, security, and ease of use. As technology and user needs continue to change, these solutions will play a critical role in ensuring the security and privacy of communication over the internet.

References

- [1] Rimeikis, A. (2022) What is openvpn and what does it have to do with your VPN?, Surfshark. Available at: <https://surfshark.com/blog/what-is-openvpn>
- [2] Community resources (2022) OpenVPN. Available at: <https://openvpn.net/community-resources/>
- [3] Donenfeld, J.A. (no date) Protocol Cryptography, WireGuard. Available at: <https://www.wireguard.com/protocol/>
- [4] IPsec vpns vs. SSL vpns | cloudflare (no date) CloudFlare. Available at: <https://www.cloudflare.com/learning/network-layer/ipsec-vs-ssl-vpn/>
- [5] Loshin, P. (2021) What is IPsec (internet protocol security)?, TechTarget. TechTarget. Available at: <https://www.techtarget.com/searchsecurity/definition/IPsec-Internet-Protocol-Security>
- [6] Molenaar, R. (2020) IPsec (internet protocol security), NetworkLessons.com. Available at: <https://networklessons.com/cisco/ccie-routing-switching/ipsec-internet-protocol-security>
- [7] Phifer, L. (2019) Choosing between an SSL/TLS VPN vs. IPsec VPN: TechTarget, Security. TechTarget. Available at: <https://www.techtarget.com/searchsecurity/feature/Tunnel-vision-Choosing-a-VPN-SSL-VPN-vs-IPSec-VPN>
- [8] Samuel (2022) Let's talk about wireguard, TheStaticTurtle's Blog. TheStaticTurtle's Blog. Available at: <https://blog.thestaticturtle.fr/lets-talk-about-wireguard/>
- [9] What is an SSL VPN? (no date) Fortinet. Available at: <https://www.fortinet.com/resources/cyberglossary/ssl-vpn>
- [10] Wireguard: Next generation kernel network tunnel (no date) WireGuard. Available at: <https://www.wireguard.com/papers/wireguard.pdf>
- [11] About IPSec Algorithms and Protocols (no date) Available at: https://www.watchguard.com/help/docs/help-center/en-US/Content/en-US/Fireware/mvpn/general/ipsec_algorithms_protocols_c.html
- [12] IPsec vs. Wireguard (no date) Tailscale. Available at: <https://tailscale.com/compare/ipsec/>
- [13] OpenVPN vs. Tailscale (no date) Tailscale. Available at: <https://tailscale.com/compare/openvpn/>
- [14] End to End Encryption VS. VPN: Differences and Which is More Secure (no date) iTopVPN. Available at: <https://www.itopvpn.com/blog/end-to-end-encryption-1224>
- [15] Lutkevich, B. and Bacon, M. (2021) What is end-to-end encryption (E2EE) and how does it work?, Security. TechTarget. Available at: <https://www.techtarget.com/searchsecurity/definition/end-to-end-encryption-E2EE>
- [16] Site to site VPN routing explained in detail (2021) OpenVPN. Available at: <https://openvpn.net/vpn-server-resources/site-to-site-routing-explained-in-detail/>
- [17] Klimas, M. (2022) What is VPN encryption and how does it work?, Surfshark. Available at: <https://surfshark.com/blog/vpn-encryption>
- [18] End-to-end encryption explained (2022) NordVPN. Available at: <https://nordvpn.com/pt/blog/what-is-end-to-end-encryption/>
- [19] Dunlap, S. (2023) Site to site VPN: How it works and do you need one?: Impactnbs;, Impact Networking. Available at: <https://www.impactmybiz.com/blog/blog-site-to-site-vpn/>
- [20] Kili, A. (2022) Aaron Kili, How to Set Up IPsec-based VPN with Strongswan on Debian and Ubuntu. Available at: <https://www.tecmint.com/setup-ipsec-vpn-with-strongswan-on-debian-ubuntu/>
- [21] Camisso, J. and McGregor, M. (2022) How to set up WireGuard on Debian 11, DigitalOcean. Available at: <https://www.digitalocean.com/community/tutorials/how-to-set-up-wireguard-on-debian-11>
- [22] Wireguard VPN - site to site (no date) Ubuntu. Available at: <https://ubuntu.com/server/docs/wireguard-vpn-site2site>
- [23] Ludwig, J. (2020) Wireguard site to site configuration, Pro Custodibus. Available at: <https://www.procustodibus.com/blog/2020/12/wireguard-site-to-site-config/>
- [24] Anton (2023) Configure site-to-site VPN with openvpn, OpsDocks. Available at: <https://opsdocks.com/posts/configure-site-to-site-openvpn/>
- [25] Establish communication between Virtual Machines (2022) OpenClassrooms. Available at: <https://openclassrooms.com/en/courses/7163136-set-up-virtual-machines-using-virtualbox-and-vsphere/7359231-establish-communication-between-virtual-machines>
- [26] Ellingwood, J. (2017) How to create a point-to-point VPN with wireguard on ubuntu 16.04, DigitalOcean. DigitalOcean. Available at: <https://www.digitalocean.com/community/tutorials/how-to-create-a-point-to-point-vpn-with-wireguard-on-ubuntu-16-04>
- [27] Drake, M., Ellingwood, J. and Shultz, K. (2022) How to set up an openvpn server on Debian 11, DigitalOcean. DigitalOcean. Available at: <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-debian-11>
- [28] Claro, E. (2020) Site-to-site IPsec VPN with Virtualbox and Openswan (with video), EDUARDO CLARO. Available at: <https://clarodba.wordpress.com/2020/06/01/site-to-site-ipsec-vpn-with-virtualbox-and-openswan-with-video/comment-page-1/>
- [29] Ludwig, J. (2020) Wireguard site to site configuration, Pro Custodibus. Available at: <https://www.procustodibus.com/blog/2020/12/wireguard-site-to-site-config/>