Multi-Tunnels: Implementation, configuration and practical assessment of network tunnels with multiple security layers

Ricardo Jorge Pimenta Lopes
Instituto Superior Técnico, Universidade de Lisboa, Portugal

***Abstract*—Computer network communication is at the foundation of how the modern world works. Whether for personal or organisational use, network communication connects people and machines. However, most of the time this communication needs to use a public network infrastructure in order to connect the two points of communication. When leaving the sphere of the user, this communication becomes vulnerable to a diversity of attacks, either by passive listening of the communication, or by active interference in the content and flow of the communication. This is where transport layer security protocols like TLS (Transport Layer Security) play a key role, as they can guarantee confidentiality, integrity, freshness and authenticity of the communication. In this paper, an extended implementation of TLS called MultiTLS is specifically addressed, which allows composing multiple TLS channels for enhanced security. By using multiple cipher suites, a vulnerability exploited in a given algorithm does not compromise the message because there are others, thus becoming vulnerability tolerant. The performance of this tool was also evaluated in order to prove the viability of its use in various personal and corporate contexts.**

Keywords: Transport Layer Security, Security, Secure Communication, Confidentiality, Tunnel, Encapsulation, Virtual Private Network

I. Introduction

From early on, the need to guarantee safe communications in various fields of society became evident. This evidence gave rise to the term cryptography, as a product of the need for secure network communication. Although cryptographic algorithms are becoming increasingly sophisticated, it is still not possible to guarantee that a given algorithm is unbreakable, therefore it is also not possible to guarantee the total security of a communication.

It is possible, however, to add layers of security to a communication, such as the use of tunnels, a case that will be analysed in this article. Based on the work carried out by Moura R. [1], we will analyse how the use of a tool called MultiTLS can contribute to a more secure communication between two points, even when it is necessary to use a public network. This tool was developed in the context of communication with enhanced security through multiple encrypted channels in a client/server architecture.

The implementation of MultiTLS required, firstly, the updating of the dependencies used by the tool, so that it could work optimally in the Ubuntu 20.04 LTS operating system, for which various modules were also created to automate and facilitate the use of the tool. Subsequently, an interface was developed in order to allow the user to control the different modules in a simpler and more intuitive way.

The evaluation of MultiTLS behaviour was developed through the application of performance tests, using two machines on which the tool was executed. With a machine in client mode and another machine in server mode, after establishing the communication between them, the sending time of files, which are characterized by their distinct size, was measured, also using different configurations of MultiTLS.

II. Secure communication: protocols, tunneling and VPN

There are several protocols used to establish secure communications, such as the set of protocols that form IPsec (Internet Protocol Secury); the TLS (Transport Layer Security) protocol; and also the SSH (Secure Shell Protocol) protocol. It is also an example the use of tunneling in the PPTP and L2TP protocols and the communication between two Ipv6 routers through an Ipv4 network as well as the use of VPN in several application scenarios.

IPsec, constituting a set of protocols, enables secure communication over the Internet. Based on the standards developed by IETF, it guarantees confidentiality, integrity, authenticity and data freshness, revealing itself as a very viable VPN tool [2]. In each IPsec operation mode, transport mode and tunnel mode, two distinct protocols can be used: AH (Authentication Header) and ESP (Encapsulation Security Protocol). Both protocols provide authentication, integrity and freshness, however, while AH does not define mechanisms to guarantee the confidentiality of the messages exchanged, ESP already allows messages to be encrypted, guaranteeing the confidentiality of the data [3]. NAT firewalls (Network Address Translation) constitute an obstacle to the use of IPsec, since they make it impossible to use IPsec AH in both modes and IPsec ESP in transport mode. Although there is the option of using IPsec in tunnel mode, and this is the option of IPsec that provides more security, it also brings a cost that is the increase of the overhead.

The TLS (Transport Layer Security), being a successor of SSL (Secure Sockets Layer), was developed in the context of security assurance in a communication over the network in a client/server architecture. This protocol has now reached its most recent version, TLS 1.3, which guarantees the authenticity, confidentiality and integrity of the connection. TLS is composed by several protocols: TLS Record Protocol; TLS Handshake Protocol; Change Cipher Spec Protocol; and Alert Protocol.

The TLS Record Protocol is used by the TLS Handshake, TLS Alert Protocol, TLS Change Cipher Spec Protocol and the application data [4] and "allows to develop mechanisms to send and receive messages" [1]. The Handshake Protocol is responsible for negotiating the session which consists of the following: a session identifier (chosen by the server), the certificates (x509 standard), the compression algorithm used to originate the TLS Compressed blocks in the TLS Record Protocol, the cipher specifications (MAC and cipher algorithm used in the TLS Record Protocol to originate the TLSCiphertext), a master secret (shared between client and server) and the flag is resumable which indicates whether the session can be used to initiate new [4] connections. The Change Cipher Spec Protocol consists of a message ciphered and compressed according to the current state of the connection, to signal a change in the negotiated cipher set. The Alert Protocol sends an alert message that, depending on the severity, can be of type warning or fatal (warning/fatal). These types of messages, like the others, are encrypted and compressed based on the current connection state [4].

On the other hand, from the need to create a tool that could administer systems and transfer files securely over non-secure networks, SSH was developed. This protocol replaced some tools like telnet, ftp, FTP/S, etc., although some of these are still in use [5]. SSH consists of three main components: the Authentication Protocol [7], the Transport Protocol [8] and the Connection Protocol [9], and these provide SSH with a great ability to perform secure communications over encrypted tunnels.

Tunneling is a tool that helps to send information securely, so that only the recipients of such information have access. However, for the connection to be successfully established it is fundamental that both parties understand and use the same protocol. In a tunneling protocol the message is encapsulated inside the datagram of another message, thus allowing it to be sent in a secure way. This characteristic in this type of protocol makes it possible to send data between two private networks, using a public network infrastructure.

PPP or Point-to-Point Protocol is a layer two protocol (data-link) used to establish a direct connection between two nodes, and allows the encapsulation of multiple layer three protocols. PPP Encapsulation defines how network layer packets are encapsulated in the PPP frame. It requires three fields: protocol, information and padding. The protocol field allows you to identify the encapsulated protocol in the information field. The information field can be filled by an arbitrary number of bits (padding) up to the Maximum Receive Unit (MRU) *moura:ist*. The PPTP (Point-to-Point Tunneling Protocol) protocol is one of the oldest VPN protocols still in use today in the context of securely exchanging PPP (Point-to-Point Protocol) messages over the network [10]. L2TP or Layer 2 Tunneling Protocol, is a VPN protocol that has its origins in two protocols: Layer 2 Forwarding Protocol (L2F) and PPTP. Like PPTP, this protocol is used to send PPP traffic through a tunnel, thus inheriting its [11] authentication and cipher methods. In L2TP there are two agents: the LAC (L2TP Access Concentrator) and the LNS (L2TP Network Server) [12]. Tunneling tools are also used so that devices with older protocols can communicate with devices supporting newer protocols, such as the Internet Protocol (IPv4 vs IPv6).

A VPN or Virtual Private Network is a connection made by a device over a public network to a private network. This connection is made using tunneling and encryption techniques that ensure that sensitive data can be transmitted securely, preventing unauthorised people from having access to that information [13]. When it comes to VPN there are several types to consider: Machine-to-Network; Network-to-Network; and Machine-to-Machine.

III. MultiTLS

The establishment of encrypted tunnels guarantees the security of a communication through an infrastructure where security is not assured. However, there may be the need to guarantee a reinforced security, and it is in this sense that the MultiTLS tool was developed. With the aim of increasing the security level of a communication, MultiTLS is a midleware with several levels of protection, which offers security by using the encapsulation of several TLS tunnels. The communication is protected by several different ciphers, one for each TLS tunnel. Thus, k * TLS tunnels will be established, where $k > 1$, in case k-1 TLS tunnels are compromised the communication remains secure [1]. MultiTLS also has the advantage that it is not necessary to change the source code of the applications/programs, since all the communication carried out will be encrypted through multiple TLS tunnels, so the tool guarantees security regardless of the application/programme being used.

TUN interfaces allow MultiTLS to create multiple virtual network interfaces. It is through the TUN interfaces that MultiTLS performs the encapsulation of the various tunnels. These interfaces act at OSI level 3, and these devices can be used to establish VPN communications, since they allow the responsible software to encrypt the information before it is sent. MultiTLS uses several TUN interfaces, since each interface will allow establishing a TLS tunnel that will be encapsulated by the TLS tunnel of the next TUN interface. On the other hand, MultiTLS uses OpenSSL as a dependency, which allows performing all the cryptographic part, from creating and signing client and server certificates to the development of message ciphering. Meanwhile, the Socat dependency allows MultiTLS to establish several tunnels. This tool allows the transfer of data between two independent channels, being responsible for creating the TUN interfaces and the use of OpenSSL. That is, it is through Socat that the tunnel between the client-side TUN interface and the server-side TUN interface is established, using the OpenSSL implementation in order to secure the connection.

That said, the big advantage of Multi TLS is its tolerance to vulnerabilities. To this end, the tool uses a different cipher set for each TLS tunnel created. The implementation of MultiTLS allows the user to initiate a connection using up to four encapsulated TLS tunnels. In the present research, four cipher sets based on the work of Moura R were used. [1]:

1) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305;
2) TLS_RSA_WITH_AES_128_CCM8;
3) TLS_DHE_DSS_WITH_CAMELIA_256_SHA256;
4) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.

To successfully establish a communication through the MultiTLS tool, it is necessary to guarantee some configurations. At first, the MultiTLS client uses port 4040 to send the information that will be used to establish the encrypted communication. This information includes: its IP address; the number of tunnels to be considered in the MultiTLS communication to be established; the port in which that communication will be made; and the client certificate used in the first tunnel. On the server side, it receives the information on port 4040 and sends its certificate for the first tunnel. The client receives the server's certificate at port 4040. Due to these initial negotiations, it is necessary to configure any firewalls that may interfere with the communication in order to accept inbound and outbound data flow to port 4040. Once the initial negotiations are finished, the tool can now establish k * tunnels (where k is the number of tunnels previously defined by the user and less than or equal to four). The first tunnel is established using the port indicated by the user when starting the program on the client side. For the remaining tunnels the port number used will be incremented from the port initially indicated by the user. Due to this property of the MultiTLS tool, it is only necessary to take into account that no firewall blocks the data flow to the port used to establish the first tunnel, since the other tunnels use the structure previously created by the first tunnel. It should also be noted that the number indicated by the user for the first port must be chosen and configured at the firewall level before initiating any communication through the MultiTLS tool. Once these configurations are made, conditions are met to establish communication between client and MultiTLS server on the same network. However, if it is necessary to carry out this communication between different networks it is necessary to configure the port forwarding. Port forwarding is port forwarding or mapping in order to redirect a communication request from one combination of IP address and port to another as the packets traverse a network gateway, such as a router [14]. Therefore, it is possible through a gateway to redirect connection requests coming from an external IP:port set to an IP and port belonging to a machine on the internal network. With the port forwarding configured in the gateways of both networks, the connection request from the client located on network 1 reaches the gateway of network 2, which in turn knows that a message coming from that IP address for that port must be delivered to the machine where the MultiTLS server is located and vice versa. Through these configurations it will be possible to establish secure communications through the MultiTLS tool, whether the client/server pair is in the same network or in different networks.

To use the MultiTLS tool it is necessary to run the tool via a terminal at the two points to be interconnected, and after execution, the program presents a menu to the user. On first execution, the user must select the first option in order to start the installation. In this step, the tool's dependencies will be installed, such as OpenSSL and Socat. This process must be carried out in both points which are to be interconnected. On finishing the installation phase, the user is redirected to the previous menu. At this stage, the user must start the tool in Server Mode at one point and Client Mode at the other. At the Client Mode point you will need to indicate the IP address of the server, the port to use and the number of tunnels you want to encapsulate (this number should be between 1 and 4). The initial settings are only necessary on the client side, because when establishing the connection with the server, it also sends the information about the port and the number of tunnels selected. On the server side, the server receives the information from the client and establishes the connection based on the settings received. Once the user completes this step, the tool creates the TLS tunnels and interconnects the two points establishing an enhanced security communication.

IV. Case studies

Currently, there are several situations where the security solution consists in the use of encrypted channels. These can be used to reinforce the security level of a communication or even to secure mechanisms that would otherwise present an obsolete security level.

In order to contextualise the use of MultiTLS in practical scenarios, four case studies were defined where the use of a tool such as MultiTLS can present a great advantage in terms of security:

1) Secure communication between areas of operation;
2) Secure communication between two cloud solutions;
3) Secure communication between the employee and the organisation's network;
4) Secure communication between legacy applications.

In the case of secure communication between areas of operation (Network-to-Network) it will be necessary to configure the MultiTLS tool in both areas of operation, which will function as reverse proxy. In this situation, it would be the responsibility of the organisation to perform the configuration of the tool in both areas of operation. Through the MultiTLS tool, it is possible to establish a secure connection through multiple encrypted TLS channels, which reinforce the security level between two areas of operation of an organisation (ex: buildings, offices, etc). In this way, despite being geographically distant, it is possible to perform data exchange securely between both areas of operation. In this study, firstly, Area 1 and Area 2 want all communication between them to be done in a more secure way due to the sensitivity of the information that needs to be exchanged between these areas. Next, Area 1 and Area 2 establish through MultiTLS, multiple TLS channels with different cipher types to provide security with diversity. Finally, traffic between areas is redirected through these channels established by the MultiTLS tool, thus adding a strong security factor.

In the next case study, secure communication between two cloud solutions, it is intended to strengthen the security in the communication between two cloud solutions, whether they are from the same vendor, or from different vendors. Similarly to the case study above described, it will also be necessary to configure a machine in each cloud solution that will serve as a reverse proxy to the remaining assets. The MultiTLS tool will be configured on both machines in order to establish the TLS channels through which communication will be redirected. The use of a tool such as MultiTLS, adds an extra security factor, which enables the existence of a system composed of several solutions cloud (cloud-of-clouds) with a high level of security of the exchanged information. In this study, firstly, it is intended to establish a communication with enhanced security between two cloud solutions. Next, Cloud 1 and Cloud 2 establish through MultiTLS, multiple TLS channels with different cipher types. Finally, traffic between cloud solutions is redirected through the channels established by the MultiTLS tool, thus adding a strong security factor.

The case study secure communication between the employee and the organization network (Machine-to-Network), intended to study the secure communication between an employee, who is outside the organization network, and the internal network of the organization. It will be the organisation's responsibility to guarantee the availability of the service (MultiTLS tool in server mode), in order to accept connection requests. On the employee's side, he/she shall configure the MultiTLS tool in client mode and establish the connection with the server. This study intends to portray the situation of an employee who is working outside the office and, therefore, is not protected by his organisation's infrastructure. However, due to the sensitivity of the information exchanged, there is a need to strengthen the security of the communication. Through the MultiTLS tool the employee can establish a secure connection between his machine and the network of his organisation. For this case study it will be necessary to configure MultiTLS on the employee's machine, and on the organisation's side it must have a machine/server to accept the communication requests from MultiTLS. In this study, firstly, the employee intends to establish a security-enhanced communication by VPN to the organisation's network. Subsequently, the collaborator establishes through MultiTLS, multiple TLS channels with different cipher types that are selected automatically and transparent to the user. Finally, the traffic between the collaborator and the organisation is redirected through these channels established by the MultiTLS tool, thus adding enhanced security to the VPN connection.

The last case, secure communication between legacy applications (Machine-to-Machine), focused on the communication between legacy applications and how to make it secure in critical environments. For this case it will be necessary that both machines run an instance of the MultiTLS tool. Either machine can run in client or server mode, however for the connection to be possible the machines must operate in different modes, i.e. one in server mode and the other in client mode. The use of the MultiTLS tool in this case study, allows legacy applications with known security vulnerabilities to be used securely. For this case study, it will be necessary to configure the MultiTLS tool on the machines where these applications that wish to communicate are located. The use of MultiTLS allows communication to be carried out in a secure manner by encapsulating the message of the legacy application through secure cryptographic protocols. A more concrete example would be the interconnection of an application server with a database that does not support a recent version of the TLS protocol. For this case study, firstly, Host 1 and Host 2 intend to establish a connection through an application that uses legacy protocols (e.g. SSL 2.0, 3.0 and TLS 1.0) and therefore not secure due to there being known vulnerabilities. Then, Host 1 and Host 2 establish through MultiTLS, multiple TLS channels that are presented as a tunnel. Finally, the traffic between the two is redirected through these channels established by the MultiTLS tool, thus making it feasible to use the legacy application.

V. Evaluation

As already explained, a set of real scenarios were prepared in order to analyse the behaviour of the MultiTLS tool in those usage contexts.

When performing the tests, two machines with the Ubuntu 20.04 LTS operating system installed and located on the same network were used. Both machines have similar computational power and configurations, with both having a four-core processor and eight gigabytes of RAM. The MultiTLS tool was configured on the machines (as well as its dependencies), in client and server mode respectively. After installing the machines and the MultiTLS tool on each host, the process of configuring the tools necessary to perform the tests was initiated.

In the machine-to-machine case study, the MultiTLS tool is used to make feasible the use of legacy applications and, therefore, with known vulnerabilities, thus making them safe to use in a real context. More specifically, FTP (File Transfer Protocol) [15] was used, which, through its client/server architecture, has the ability to establish a connection between two points, which can be used to transfer files and perform other operations. However, due to its age, FTP in its original form does not have any type of encryption during communication, and this is the reason why it is less and less used, especially in a corporate context. This characteristic that makes it insecure, at the same time makes it an excellent example of how a tool like MultiTLS can play a fundamental role, guaranteeing the encryption process of messages and making FTP a viable alternative.

Before carrying out the tests, it was necessary to install and configure the FTP client and server on the respective machines. Next, four files of different sizes were considered:

- 25MFile;
- 50MFile;
- 75MFile;

- 100MFile.

It was taken into account that the files should be neither too small, as it would hinder the data collection process, nor too large, as it would hinder the process of carrying out several measurements. After some initial attempts it was concluded that a file size of 25 MB would meet both these requirements. The sizes of the remaining files are multiples of the first value. This results in a constant increase in file size which facilitates data analysis.

The tests performed consisted of transferring the different files (25MB, 50MB, 75MB and 100MB) and measuring the transfer time for different numbers of tunnels. Initially, as a form of reference, a test was performed for k = 0, i.e., the measurements were performed without using any tunnel, just a normal FTP communication. Next, the same procedure was performed for k = 1, k = 2, k = 3 and k = 4, where k represents the number of tunnels used by the MultiTLS tool.

In order to minimise the impact of possible network disturbances, all the results obtained were collected at the same time of the day, under similar conditions. Another aspect that was taken into account was the representativeness of the data. In order to guarantee that the data sample collected was representative, the standard deviation of the data obtained was calculated. After collecting the samples it was found that the standard deviation for all cases considered was less than one second. Considering that the network may sometimes suffer some disturbances, this standard deviation is considered a very acceptable value. Finally, an average of the obtained measurements was made in order to be analysed.

A. Results

The results obtained through the procedures described above will now be presented in the form of tables. These are organised according to the number of tunnels, and five tables are presented (k = 0, k = 1, k = 2, k = 3 and k = 4). Each table also has five columns: the first indicates the order in which the data was inserted, and the remaining four indicate the value of the measurements for the different files considered. Finally, each table also indicates the average transfer time for each file. The tables present in this section represent only a summarised version of the original ones.

In table I, the data referring to the transfer of the files using only FTP (k = 0) are represented.

k = 0				
	25MFile	50MFile	75MFile	100MFile
1º	0.08 s	0.20 s	0.24 s	0.31 s
2º	0.08 s	0.18 s	0.24 s	0.33 s
3º	0.07 s	0.15 s	0.24 s	0.26 s
4º	0.08 s	0.15 s	0.23 s	0.44 s
5º	0.07 s	0.14 s	0.23 s	0.28 s
	...	...	...	...
Mean	0.08 s	0.17 s	0.22 s	0.33 s
σ	0.01 s	0.02 s	0.01 s	0.06 s

TABLE I: Test values for k = 0

Using only FTP (k = 0), it was possible to verify that the sending of the files was practically instantaneous. Next, the same test was performed, however, this time for k = 1, that is, using the MultiTLS tool with only one tunnel configured. The results obtained can be seen in table II. Through the

k = 1				
	25MFile	50MFile	75MFile	100MFile
1º	2.90 s	3.78 s	6.93 s	9.86 s
2º	2.25 s	3.63 s	6.63 s	9.81 s
3º	3.25 s	2.86 s	7.50 s	11.90 s
4º	2.58 s	2.87 s	8.77 s	8.77 s
5º	1.53 s	3.59 s	5.44 s	8.09 s
	...	...	...	...
Mean	2.87 s	3.30 s	7.39 s	9.54 s
σ	0.96 s	0.88 s	0.96 s	0.92 s

TABLE II: Test values for k = 1

observed data it was possible to verify that when configuring a connection with only one tunnel, an increase in the average times is already noticeable. The remaining data, referring to k = 2, k = 3 and k = 4, that is, the MultiTLS tool configured with two, three and four encapsulated tunnels, can be observed in the tables III, IV and V respectively.

k = 2				
	25MFile	50MFile	75MFile	100MFile
1º	3.00 s	8.38 s	14.41 s	19.16 s
2º	4.14 s	7.28 s	16.09 s	20.38 s
3º	2.07 s	7.63 s	13.92 s	21.38 s
4º	5.81 s	7.04 s	15.78 s	19.98 s
5º	3.99 s	6.16 s	13.24 s	21.12 s
	...	...	...	...
Mean	3.41 s	7.70 s	14.46 s	20.26 s
σ	0.81 s	0.94 s	0.90 s	0.95 s

TABLE III: Test values for para k = 2

k = 3				
	25MFile	50MFile	75MFile	100MFile
1º	7.61 s	16.95 s	19.42 s	30.48 s
2º	6.81 s	14.89 s	20.39 s	29.51 s
3º	5.93 s	15.27 s	20.05 s	28.43 s
4º	5.48 s	15.54 s	19.51 s	30.76 s
5º	8.13 s	17.71 s	20.05 s	29.78 s
	...	...	...	...
Mean	7.20 s	16.00 s	20.28 s	30.17 s
σ	0.95 s	0.93 s	0.93 s	0.96 s

TABLE IV: Test values for k = 3

k = 4				
	25MFile	50MFile	75MFile	100MFile
1º	10.95 s	19.91 s	26.68 s	39.35 s
2º	10.44 s	18.54 s	27.11 s	38.83 s
3º	9.07 s	19.66 s	26.31 s	40.08 s
4º	10.13 s	18.91 s	26.54 s	41.02 s
5º	9.00 s	17.71 s	20.05 s	29.78 s
	...	...	...	... s
Mean	9.33 s	19.33 s	27.18 s	39.64 s
σ	0.90 s	0.84 s	0.99 s	0.96 s

TABLE V: Test values for = 4

VI. Discussion

Regarding the analysis of results, the graph of the figure 1, was obtained through the data, collected in the experiments performed, which can be observed in the tables I, II, III, IV and V. This graph, allows us to make a comparison of the time involved in the transfer of files, when using files of different sizes and different numbers of tunnels. That is, through the graph of the figure 1 it is possible to observe how the number of tunnels, used in the configuration of the MultiTLS tool, and the size of the data to be transferred during the communication, can affect the performance of MultiTLS. Observing the ordinate axis, it is possible to verify that it is represented in seconds (s). This axis, represents the transfer time of each file. As for the abscissae axis, it is divided by the size of the files used in the set of experiments (25 MB, 50MB, 75 MB and 100 MB) and, consequently, for each file five different values are presented. These values correspond to the number of tunnels used, during the experiments, to download each file, where:

- Zero means that no tunnel was considered and therefore only FTP was used in the file transfer;
- Number one represents the file transfer using the Multi-TLS tool with only one tunnel configured;
- Number two means that 2 tunnels were used during the file transfer;
- In number three the file upload is performed with three tunnels configured in the MultiTLS tool;
- Finally, in number four the MultiTLS tool is used with four tunnels configured.

Analysing the graphic of the figure 1, it is possible to verify that the greater the size of the file and the number of tunnels used, the greater the transfer time will be. Something that would be expected, considering that not only the size of the file itself was increased but also the overhead caused by adding tunnels to the communication. Thus, it is possible to conclude that the size of the file to be transferred and the number of tunnels used to protect the connection, have a direct impact on the performance of the MultiTLS tool. Another phenomenon that can be observed, is that as the file size increases there is also an increase in the difference of the times for different numbers of tunnels. Which leads to the conclusion that the number of tunnels chosen for communication has a more significant impact on transfer time the larger the file size.

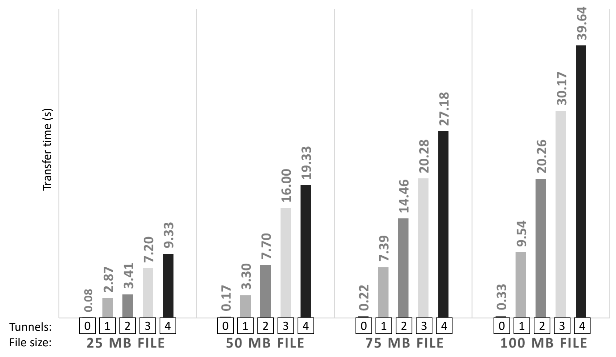

Fig. 1: Chart - Test results

That said, it would be easy to fall into the error of disregarding the usefulness of a tool like MultiTLS, based only on the high transfer times for larger files in cases where the highest number of tunnels are used. However, it should be clarified that everything depends on the use case in question. A tool will only be effective if the work performed is appropriate for its use. MultiTLS, being a tool, is no exception. If the goal is to send the file as quickly as possible, it might be better to go for another alternative. What MultiTLS does offer, however, is the ability to securely send extremely sensitive information due to its fault-tolerant property.

As an example, a company X wants to send the following document:

- File type: .txt;
- Number of characters: around 595 000 (equivalent to 100 pages);
- File size: 0.582 MB;
- Classification: Highly Confidential.

If a 25 MB file in the tests performed was transferred in an average of 9.33 seconds for k = 4, the text file of company X, classified as Highly Confidential and equivalent to 100 pages (0.582 MB) would be transferred via MultiTLS in a very short period of time, considering similar conditions to those encountered when the data was collected.

With this example as a reference, it is possible to see that it is not enough to take into account the size of the information and transfer time. In this case, it is also important to understand what it means, and what the impact is if that information is intercepted by unauthorised individuals. Based on all factors, it is considered that MultiTLS is a very useful tool whenever the goal is to reinforce the security factor of a communication, or even add this factor where it does not exist in the first instance.

Additionally, the average transfer speed of each file was calculated for the different numbers of tunnels. These values, present in the table VI, were calculated from the average values of the transfer times already presented.

	Transfer speed (MB/s)			
	25MFile	50MFile	75MFile	100MFile
k = 0	312.50 MB/s	294.11 MB/s	340.90 MB/s	303.03 MB/s
k = 1	8.71 MB/s	15.15 MB/s	10.14 MB/s	10.48 MB/s
k = 2	7.33 MB/s	6.49 MB/s	5.18 MB/s	4.93 MB/s
k = 3	3.47 MB/s	3.12 MB/s	3.69 MB/s	3.31 MB/s
k = 4	2.67 MB/s	2.58 MB/s	2.89 MB/s	2.52 MB/s

TABLE VI: Transfer speed

The 2 graph, created from the data in the VI table, allows us to verify that there is a notable difference in the average speed calculated, when transferring the file, without any type of protection (k = 0), when compared with the remaining cases where tunnels were used to protect the communication (k = 1, k = 2, k = 3 and k = 4). However, it should also be noted that, although the difference in performance from k = 0 to k = 1 is evident, for the remaining cases this difference is less pronounced. This means that, when the mechanisms are used to protect the communication, they have a strong impact on its performance. This is true even for k = 1, which would

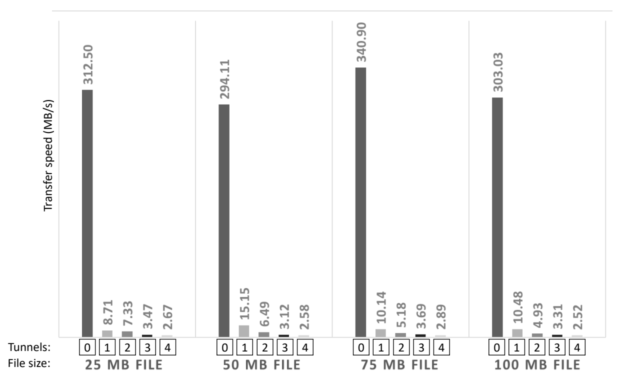

Fig. 2: Chart - Transfer speed (MB/s)

be equivalent to using only the TLS protocol. However, when more tunnels are added to the communication (k = 2 , k = 3 and k = 4), although there is a negative impact on performance, this impact is less evident when compared with the difference between the case without security (k = 0) and the use of a security level equivalent to using only the TLS protocol (k = 1).

VII. Conclusion

The cryptographic algorithms as used in communication protocols play a key role in network communication. In the course of this research it was verified the existence of a wide range of cryptographic algorithms, some being legacy protocols, therefore not so used anymore, and others more modern and used nowadays. MultiTLS, being an extended implementation of TLS, guarantees increased security through the encapsulation of several TLS tunnels, where several ciphers can be used and, in this way, allows tolerating attacks that compromise the security of one cipher but do not affect the others.

Throughout this work, several algorithms and concepts that contributed and still contribute nowadays to make communication possible in a safe way *online* were presented. It was also presented the implementation of the MultiTLS tool, including its architecture, its dependencies and how it can be easily installed and configured. Next, four case studies were presented where the security solution was to use MultiTLS and, finally, a set of experiments were carried out in order to evaluate the performance of MultiTLS. Based on the results and conclusions obtained, it was proven that a tool such as MultiTLS can be a way to ensure enhanced security in communications that require it, whether for individual use or at an organisational level. Its usefulness was also demonstrated through the application of the four case studies already mentioned. In each of them, MultiTLS revealed that its tolerance to vulnerabilities is a feature that gives it a more solid security, and that makes it evident when compared with the use of a normal VPN.

Additionally, it was also verified how MultiTLS performs with different file sizes and different numbers of tunnels, proving that these two variables influence significantly the performance of MultiTLS. The larger the file size, the greater the impact of the number of tunnels chosen on the transfer time. This increase in transfer time, when a larger number of tunnels are used, is essentially due to the cryptographic processes involved in the creation of the different tunnels.

The MultiTLS tool, is capable of providing enhanced security to communications that require it, however it should be noted that the tool still has room for evolution.

The use of MultiTLS in the enterprise context will require the existence of a user authentication mechanism. For that to be feasible, it would be necessary for the tool to have an integration mechanism that allows the provision of identity, i.e., that allows extracting from a database the necessary information about the company's users. Therefore, users would only be able to access, through MultiTLS, the company's resources through authentication, which would also allow the limitation of access to certain resources. Having said that, it would be interesting in a future work to work on these issues, since the lack of authentication is a shortcoming of the MultiTLS tool. Although, as has been shown, MultiTLS has several advantages when used, it does not provide security against unwanted users at its root. Therefore, it would be a great consolidation of the tool to guarantee that not all users who install MultiTLS have access to the other side of the communication. To have such access, it would be necessary to prove that the user is who he really says he is and that he has authorisation and, therefore, credentials to access all the functionalities of the system.

On the other hand, it could be interesting and even quite useful to work on the relationship of MultiTLS with operating systems. At the moment, this tool only supports some environments such as *Linux*, leaving aside systems like *Windows*, *Android* and *iOS*, which are very relevant systems in the world market.

References

[1] M. R., "Multitls: Secure communication channel with diversity," Master's thesis, Instituto Superior Técnico, junho 2018.

[2] CISCO, "Introduction to cisco IPsec technology," https://www.cisco.com/c/en/us/td/docs/net_mgmt/vpn_solutions_center/2-0/ip_security/provisioning/guide/IPsecPG1.html, agosto 2007.

[3] IBM, "What is the difference between the ah and esp protocols of ipsec?" https://www.ibm.com/support/pages/what-difference-between-ah-and-esp-protocols-ipsec, maio 2017.

[4] D. T. e Rescorla E., *RFC5246-The Transport Layer Security (TLS) Protocol Version 1.2*, IETF, agosto 2008.

[5] SSH, "Ssh (secure shell)," https://www.ssh.com/ssh/.

[6] Y. T. e Lonvick C., *RFC4251 - The Secure Shell (SSH) Protocol Architecture*, IETF, janeiro 2006.

[7] ——, *RFC4253 - The Secure Shell (SSH) Transport Layer Protocol*, IETF, janeiro 2006.

[8] ——, *RFC4252 - The Secure Shell (SSH) Authentication Protocol*, IETF, janeiro 2006.

[9] ——, *RFC4254 - The Secure Shell (SSH) Connection Protocol*, IETF, janeiro 2006.

[10] S. B. M. e Wagne D., "Cryptanalysis of microsoft's pptp authentication extensions (ms-chapv2)," https://people.eecs.berkeley.edu/ daw/papers/pptpv2.pdf, outubro 1999.

[11] e. a. Patel, *RFC3193-Securing L2TP using IPsec*, IETF, novembro 2001.

[12] T. et al., *RFC2661-Layer Two Tunneling Protocol "L2TP"*, IETF, agosto 1999.

[13] CISCO, “What is a vpn? - virtual private network,” https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html.

[14] J. A. Verma N, Kashyap M, *Extending Port Forwarding Concept to IOT*. India: IEEE, 2018.

[15] R. J. Postel J., *File Transfer Protocol (FTP)*, IETF, outubro 1985.