SuperTLS: A Diverse and Redundant Secure Communication Channel for Privacy in Cloud

André Joaquim
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal

Miguel L. Pardal
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal

Miguel Correia
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal

{andre.joaquim, miguel.pardal, miguel.p.correia}@tecnico.ulisboa.pt

ABSTRACT

We present superTLS, a diverse and redundant vulnerability-tolerant secure communication channel for privacy in cloud. There have always been concerns about the strength of some of encryption mechanisms used in SSL/TLS channels and some of them were regarded as insecure at some point in time. SuperTLS is our solution to mitigate the problem of secure communication channels being vulnerable to attacks due to unexpected vulnerabilities in its encryption mechanisms. It is based on diversity and redundancy of cryptographic mechanisms and certificates to provide a secure communication channel even when one or more mechanisms are regarded vulnerable. SuperTLS relies on a combination of k mechanisms/cipher suites, with k being the diversity factor and $k > 1$. Even when $k - 1$ mechanisms are regarded as insecure or considered vulnerable, SuperTLS relies on the remaining secure, diverse and redundant mechanism to maintain the channel secure. We evaluated the performance of our channel by comparing it to a normal TLS channel.

CCS Concepts

•**Networks** → **Application layer protocols;** *Network security;* •**Computer systems organization** → **Redundancy;** •**Security and privacy** → *Security protocols;* Symmetric cryptography and hash functions;

Keywords

Secure communication channels; Diversity; Redundancy; TLS; Vulnerability-Tolerance

1. INTRODUCTION

Secure communication channels are mechanisms that allow two entities to exchange messages or information securely in the Internet. A secure communication channel has three properties: *authenticity*, *confidentiality*, and *integrity*. Regarding authenticity, in an authentic channel, the messages can not be tampered. Regarding confidentiality, in a confidential channel, only the original receiver of a message is able to read that message. Regarding integrity, no one can impersonate another. The information regarding the original sender of a message can not be changed. Several secure communication channels exist nowadays, such as TLS, IPsec or SSH. Each of these examples is used for a different purpose, but with the same finality of securing the communication.

Transport Layer Security (TLS) is a secure communication channel widely used. Originally called Secure Sockets Layer (SSL), its first released version was SSL 2.0, released in 1995. SSL 3.0 was released in 1996, bringing improvements to its predecessor such as allowing forward secrecy and supporting SHA-1. Defined in 1999, TLS did not introduce major changes. Although, the changes introduced were enough to make TLS 1.0 incompatible with SSL 3.0. TLS 1.1 and TLS 1.2 are upgrades to TLS 1.0 which brought some improvements such as mitigating CBC (cipher block chaining) attacks and supporting more block cipher modes of operation to use with AES. TLS is divided in two sub-protocols, Handshake and Record, constituted by several mechanisms each. The Handshake protocol is used to establish or re-establish a communication between a server and client. The Record protocol is used to process the sent and received messages.

Internet Protocol Security (IPsec) is an Internet layer protocol that protects the communication at a lower level than SSL/TLS, which operates at the Application layer [4].

Secure Shell (SSH) is an Application layer protocol, such as SSL/TLS. SSH is a protocol used for secure remote login and other secure network services over an insecure network [5].

A secure communication channel becomes insecure when a vulnerability is discovered. Vulnerabilities may concern the protocol's specification, cryptographic mechanisms used by the protocol or specific implementations of the protocol. Many vulnerabilities have been discovered in SSL/TLS originating new versions of the protocol with renewed security aspects such as deprecating cryptographic mechanisms or enforcing security measures. Concrete implementations of SSL/TLS have been also considered vulnerable by having implementation details causing a breach of security and affecting devices worldwide.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

SuperTLS is a secure communication channel tolerant to vulnerabilities which does not rely on only one cryptographic mechanism. It is our belief that *diversity* and *redundancy* of cryptographic mechanisms and certificates can help mitigate existent vulnerabilities. In our project's context, diversity and redundancy consist in using two or more different mechanisms/cipher suites with the same objective. For example, MD5 and SHA-3 are both hash functions used to generate digests. In a real case, where MD5 has become insecure, our diverse and redundant secure communication channel relies upon SHA-3 to keep the communication secure. Using diversity and redundancy of cryptographic mechanisms, when a one of those mechanisms is successfully attacked, another mechanism is able to maintain the security and availability of the communication.

- Maybe talk a little about the project's context – the cloud and communication between clouds

SuperTLS is part of an European project named SafeCloud...

Using diversity and redundancy, SuperTLS aims at increasing security over current secure communication channels by guaranteeing a diversity factor $k > 1$, which implies having k different cipher suites with optimally k different mechanisms for each of the following:

- Key exchange;
- Authentication;
- Encryption;
- MAC: HMAC/AEAD (used for data integrity). **NOTE: In TLS, HMAC is used for CBC and stream cipher. AEAD is used for GCM and CCM (MtE)**

Although SSL/TLS supporting strong encryption mechanisms, such as AES and RSA, there are other factors than mathematical complexity that can contribute to vulnerabilities. Diversifying encryption mechanisms includes diversifying certificates and consequently keys (public, private, shared). Diversity of certificates is a direct consequence of diversifying encryption mechanisms due to the fact that each certificate is related to an authentication and key exchange mechanism.

The contributions of this paper are a new secure communication channel which uses diversity and redundancy to tolerate vulnerabilities in existent cryptographic mechanisms based on TLS 1.2 and specific evidence that diversity and redundancy can be employed without creating an excessive overhead in the communication. Proving that diversity and redundancy have a real impact on increasing security of a communication channel, while having reasonable performance and time-related costs, required a precise evaluation of our solution.

The rest of the document is organized in the following way: Section 2 presents the related work. Section 3 presents the architecture of the proposed solution. Section 4 presents the evaluation. Section 5 presents some conclusions.

2. RELATED WORK

What are the current issues/vulnerabilities in the existing systems/mechanisms?

- Brief TLS
- Mechanisms vulnerabilities
- Brief combining mechanisms OR brief diversity

3. OVERVIEW

SuperTLS is a diverse and redundant vulnerability-tolerant secure communication channel. It aims at increasing security using diverse and redundant cryptographic mechanisms and certificates, and it is based on the TLS protocol. Although being an independent secure communication channel, SuperTLS is compatible with TLS 1.2.

This project aims to solve the main problem originated by having only one cipher suite negotiated between client and server: when one of the cipher suite's mechanisms becomes insecure, the communication channels using that cipher suite may become vulnerable. Although most cipher suites' cryptographic mechanisms supported by TLS 1.2 are regarded as secure, as stated in Section 2, there is not any assurance that a governmental agency or a company with high computational power and financial resources is not able to break one of those cryptographic mechanisms in the near future.

Unlike TLS, a SuperTLS communication channel does not rely in only one cipher suite. SuperTLS negotiates more than one cipher suite between client and server and, consequently, more than one cryptographic mechanism will be used for each **phase/purpose** – key exchange, authentication, encryption and MAC.

Diversity and redundancy's first entry point in SuperTLS is in the SuperTLS' Handshake, where client and server negotiate k cipher suites to be used in the communication, where k, $k \geq 1$ is called the *diversity factor*. In an abnormal case where the diversity factor $k = 1$, it is considered that the communication channel has no diversity nor redundancy. Nevertheless, in this case, SuperTLS works as a regular TLS 1.2 channel with one cipher suite. The strength of SuperTLS resides in the fact that, even when $(k - 1)$ cipher suites become insecure, because one of its cryptographic mechanisms is insecure, our proposal remains invulnerable. The remaining secure, diverse and redundant cipher suite ensures that the communication channel is secured by remaining invulnerable. The server chooses the best combination of k cipher suites according to the cipher suites server and client have available. The choice of the cipher suites might be conditioned by the certificates of both server and client. Diversity and redundancy will also naturally be introduced in the following communication between client and server. SuperTLS uses a subset of the k cipher suites agreed-upon in the Handshake Protocol to encrypt the messages.

SuperTLS negotiates a default of $k = 2$ cipher suites, implying a diversity factor $k = 2$. While performance and management costs must be taken into account, this is the estimate of a reasonable k cipher suites to use in the communication.

NOTA1: Ver se a tese do ricardo diz algo acerca de k=2 ser o melhor

NOTA2: Será que posso afirmar que conforme o k cresce, a diversidade é cada vez menor e que 2 é a melhor relacao entre performance e diversidade?

In terms of security, our solution must tolerate all the attacks given that at least one diverse redundant mechanism of the one attacked should not to be vulnerable that attack.

On the contrary, our solution should never be vulnerable to attacks to which TLS 1.2 is not vulnerable.

3.1 Protocol Specification

SuperTLS's Handshake Protocol is similar to TLS Handshake Protocol. Messages' name are identical in order to provide easier migration and transition from TLS. Additionally, all the SuperTLS messages' names are analogous to TLS. Using this simplification, anyone who is familiarized with TLS can easily understand SuperTLS' Handshake messages and their purpose.

The messages which are diversity entry points are ClientHello, ServerHello, ServerKeyExchange, k-ServerKeyExchange, (Server and Client) Certificate, ClientKeyExchange and k-ClientKeyExchange.

The first message to be sent is called *ClientHello*. This message's purpose is to inform the server that the client wants to establish a diverse secure channel for communication. The content of this first message consist in the client's protocol version, a Random structure (analogous to TLS 1.2) containing the current time and a 28-byte pseudo-randomly generated number, the session identifier, a list of the client's cipher suites and a list of the client's compression methods, if compression is to be used.

The server responds with a message named *ServerHello*. ServerHello is a very important message as it is where the server sends to the client the k cipher suites to be used in the communication. The server sends to the client its protocol version, a Random structure identical to the one sent by the client, the session identifier and the k cipher suites chosen by the server from the list the client sent. It is also sent the compression method to use, if compression is to be used in the communication.

The server proceeds to send a *(Server) Certificate* message containing its k certificates to the client. The k chosen cipher suites are dependent from the server's certificates. Each certificate is associated with one key exchange mechanism (KEM). Therefore, the k cipher suites must use the key exchange mechanisms supported by the server's certificates.

NOTA: Devo incluir o seguinte paragrafo?

However, SuperTLS behaves normally if the server has $x, 0 < x < k$ certificates. The cipher suites to be used are chosen considering the available certificates. In this case, the diversity is not maximum due to the fact that a number of cipher suites will share the same certificate.

The *ServerKeyExchange* message is the next message to be sent to the client by the server. This message is only sent if one of the k cipher suites includes a key exchange mechanism which uses ephemeral keys, namely ECDHE or DHE. The contents of this message are the server's Diffie-Hellman ephemeral parameters. For every other $k-1$ cipher suites using ECDHE or DHE, the server sends additional ServerKeyExchange messages with additional diverse Diffie-Hellman ephemeral parameters.

NOTA: Devo tentar justificar porque é que mando mais mensagens em vez de tentar incluir tudo na mesma? Instead of computing all the ephemeral parameters and sending a larger message, the server computes one parameter at once and sends it immediately.

The remaining messages sent by the server to the client at this point of negotiation, CertificateRequest and ServerHelloDone, are identical to TLS 1.2 [2].

NOTA: O Client não precisa mesmo de enviar k certificados... alterei no esquema. o cliente envia i e o servidor envia k ou x. Rever. The client proceeds to send a *(Client) Certificate* message containing its i certificates to the server, analogous to the (Server) Certificate message the client received previously from the server.

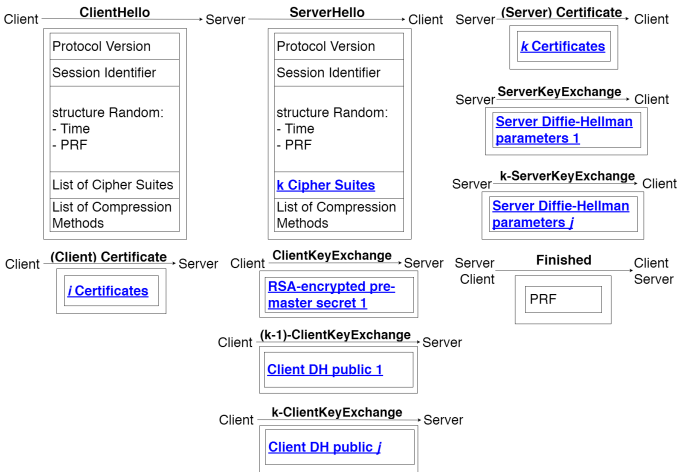

Figure 1: SuperTLS' Handshake messages using a diversity factor k. The diversity and redundancy entry points are marked in blue and underlined.

After sending its certificates, the client proceeds to send k *ClientKeyExchange* messages to the server. The content of these messages is based on the k cipher suites chosen. If $m, 0 \leq m \leq k$ of the cipher suites use RSA as KEM, the client sends m messages, each one with a RSA-encrypted pre-master secret to the server. If $j, 0 \leq j \leq k$ of the cipher suites use ECDHE or DHE, the client sends j messages to the server containing its j Diffie-Hellman public values. Even if a subset of the k cipher suites share the same KEM, this methodology still applies as we introduce diversity by using different parameters for each cipher suite being used.

The server may need to verify the client's i certificates. If they have signing capabilities, the client digitally signs all the previous handshake messages and sends them to the server for verification.

Client and server now exchange *ChangeCipherSpec* messages, alike the Cipher Spec Protocol of TLS 1.2, in order to state that they are now using the previously negotiated cipher suites for exchanging messages in a secure fashion.

The client and server, in order to finish the Handshake, send to each other a *Finished* message. This is the first message sent encrypted using the k cipher suites negotiated earlier. Its purpose is to each party receive and validate the data received in this message. If the data is valid, client and server can now exchange messages over the communication channel.

3.2 Combining diverse cipher suites

Combining cryptographic mechanisms is not trivial as not all cryptographic mechanisms are compatible with one another. As referred in Section 2, cryptographic mechanisms must be combined in a way that security is increased. For that to happen, we want to maximize diversity. In order to fulfil these constraints, some research was made to determine and quantify diversity among some cryptographic mechanisms [1].

Diversity is measured using different metrics for hash functions or public-key cryptographic functions. Hash functions' metrics include origin, year, digest size, structure, rounds and weaknesses (collisions, second preimage and preimage). After comparing several hash functions using the metrics stated above, the authors concluded that the best three combinations are the following:

- SHA-1 + SHA-3: This combination is not possible in SuperTLS. SHA-1 is regarded insecure and TLS 1.2 does not support SHA-3;
- SHA-1 + Whirlpool: This combination is not possible in SuperTLS. SHA-1 is regarded insecure and TLS 1.2 does not support Whirlpool;
- SHA-2 + SHA-3: This combination is not possible in SuperTLS. TLS 1.2 does not support SHA-3.

All the remaining suggested combinations cannot also be used because TLS 1.2 does not support SHA-3. These results have a direct impact in SuperTLS due to the fact that, being SuperTLS based on OpenSSL, and compatible with TLS 1.2, it also does not support SHA-3 nor Whirlpool. All of SuperTLS' cipher suites use either AEAD (MAC-then-Encrypt mode using a SHA-2 variant) or SHA-2 (SHA-256 or SHA-384).

Having a small range of available hash functions limits the maximum diversity factor achievable concerning hash functions. SHA-3 is relatively recent, having been selected the winner of the NIST hash function competition on 2012. In a near future, it is expected that a new TLS protocol version supports SHA-3 and makes possible the use of diverse hash functions. Nevertheless, it still possible to achieve diversity by using different variants of SHA-2 – SHA-256 and SHA-384.

Regarding public-key functions, the metrics used include origin, year, mathematical hard problems, perfect forward secrecy, semantic security and known attacks. After comparing several public-key encryption mechanisms, using the metrics stated above, the authors concluded that the best four combinations are the following:

- DSA + RSA: This combination is possible as TLS 1.2 suports both functions for *authentication*. However, TLS 1.2 specific cipher suites only support DSA with elliptic curves (ECDSA);
- DSA + Rabin-Williams: This combination is not possible. TLS 1.2 does not support Rabin-Williams;
- RSA + ECDH: This combination is possible as TLS 1.2 supports both functions for *key exchange*;
- RSA + ECDSA: This combination is possible as TLS 1.2 supports both functions for *authentication*.

Regarding authentication, although DSA + RSA is stated as the most diverse combination, TLS 1.2 preferred cipher suites use ECDSA instead of DSA. Using elliptic curves results in a faster computation and lower power consumption [3]. With that being said, the preferred combination for authentication is RSA + ECDSA.

Regarding key exchange, the most diverse combination is RSA + ECDH. Although, in order to grant perfect forward secrecy, the ECDH must be employed using ephemeral keys (ECDHE). Concluding, the preferred combination for key exchange is RSA + ECDHE.

The study did not presented any conclusions regarding symmetric-key encryption, such as AES. Therefore, considering the metrics – origin, year, and semantic security – employed for public-key encryption functions, and considering an additional metric – the mode of operation – we obtained combinations of diverse symmetric-key encryption:

- AES256-GCM + CAMELLIA128-CBC: This combination is possible as TLS 1.2 suports both functions;
- AES256-CBC + CAMELLIA128-GCM: This combination is possible as TLS 1.2 suports both functions;
- AES128-GCM + CAMELLIA256-CBC: This combination is possible as TLS 1.2 suports both functions;
- AES128-CBC + CAMELLIA256-GCM: This combination is possible as TLS 1.2 suports both functions.

Both AES and Camellia are supported by TLS 1.2 and are considered secure. The most diverse combination is AES256-GCM + CAMELLIA128-CBC. Their origin is different, they were first published in different years, they both have semantic security (as they both use initialization vectors) and the mode of operation is also different. One constraint of using this combination is that there is no cipher suite that uses RSA for key exchange, Camellia for encryption and a SHA-2 variant for MAC. Therefore, and in order to create maximum diversity, using Camellia implies using ECDHE for key exchange with Camellia.

As the cipher suites list is not very extensive, we are able to select a diverse group of k cipher suites without generating a considerable amount of overhead. As the cipher suites are presented in order of preference, the first cipher suite chosen is, ideally, the first of the list.

The optimal combination of cipher suites is:
TLS_RSA_WITH_AES_256_GCM_SHA384 +
TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256.

For key exchange, as stated above, SuperTLS will use RSA and Ephemeral ECDH (ECDHE); for authentication, it will use RSA and Elliptic Curve DSA (ECDSA); for encryption, it will use AES-256 with Galois/Counter mode (GCM) and Camellia-128 with cipher block chaining (CBC) mode; finally, for MAC, it will use SHA-2 variants (SHA-384 and SHA-256).

Using this combination of cipher suites, maximum diversity is achieved using a diversity factor $k = 2$. The least diversified part of the communication is the MAC, due to the fact that TLS 1.2 does not support SHA-3 for now.

3.3 Implementation

SuperTLS is based on OpenSSL v1.0.2g. Creating a secure communication channel from scratch is a risk because it could have vulnerabilities that were not detected and which could cause a breach of security. Additionally, creating a new secure communication channel, and consequently a new API, would create many entrance barriers to anyone who wanted to use our new communication channel. Therefore, we chose to implement SuperTLS based on OpenSSL to avoid the maximum amount of vulnerabilities originated from coding errors and to make it easier to migrate from OpenSSL to SuperTLS, as the SuperTLS' API contains the

same functions as the OpenSSL's API plus the SuperTLS' specific calls/functions.

Although being based on OpenSSL, SuperTLS is not compatible with OpenSSL. Due to its diversity and redundancy characteristics, SuperTLS can not connect to a OpenSSL server or client because, even though it is possible to use SuperTLS with just one certificate, even then SuperTLS will aim maximizing diversity with the purpose of increasing security.

Besides this fact, SuperTLS does not support all of OpenSSL cipher suites, such as PSK and SRP cipher suites.

NOTA: Explicar aqui porque é que escolhemos $k = 2$

In order to establish a SuperTLS communication channel, additional functions are required to fulfil the requirements of SuperTLS, such as loading two certificates and corresponding private keys. These functions have a similar name of the ones belonging to the OpenSSL API, to minimize the learning curve. The most relevant functions regarding the setup of the channel are the functions that allow to load the second certificate and private key and allow to check if the second private key corresponds to the second certificate.

Regarding the SuperTLS Handshake, we opted to send $k, k = 2$ ServerKeyExchange, and ClientKeyExchange, messages instead of sending one single ServerKeyExchange, and one single ClientKeyExchange, each one with several parameters. This is due to the fact that it is easier to understand and to maintain the code. If k needs to be increased, just send an additional message instead of changing the code related to sending and retrieving ServerKeyExchange and ClientKeyExchange messages.

The signing and encryption ordering is also very important for SuperTLS. Figure 2 shows the ordering for one cipher and one MAC, i.e., the OpenSSL implementation. We opted to keep the signing prior to the encryption, regarding both ciphers. The first possibility was, ordered from first to last:

1. Apply the second MAC to the message m;
2. Apply the first MAC to the message m;
3. Encrypt to the message m plus both MACs with the first cipher;
4. Encrypt the first ciphertext with the second cipher.

This caused some issues regarding the MAC. Applying two MACs in a row, using the same message and the same hash function, which can happen due to TLS' cipher suites using mostly SHA-2 variants, would possibly generate two equal MACs. This is not a desired outcome. Due to this fact, a second possibility was proposed, also ordered from first to last:

1. Apply the first MAC to the message m;
2. Encrypt the first message and its MAC with the first cipher;
3. Apply the second MAC to the first ciphertext;
4. Encrypt the first ciphertext and its MAC with the second cipher.

In both cases, the message and the MACs are encrypted with both ciphers. The second possibility was the chosen to apply in SuperTLS because in this case there is not a chance that both MACs are identical, if the hash function used is secure (SHA-2 is considered secure). Figure 3 demonstrates the final ordering of SuperTLS communication.

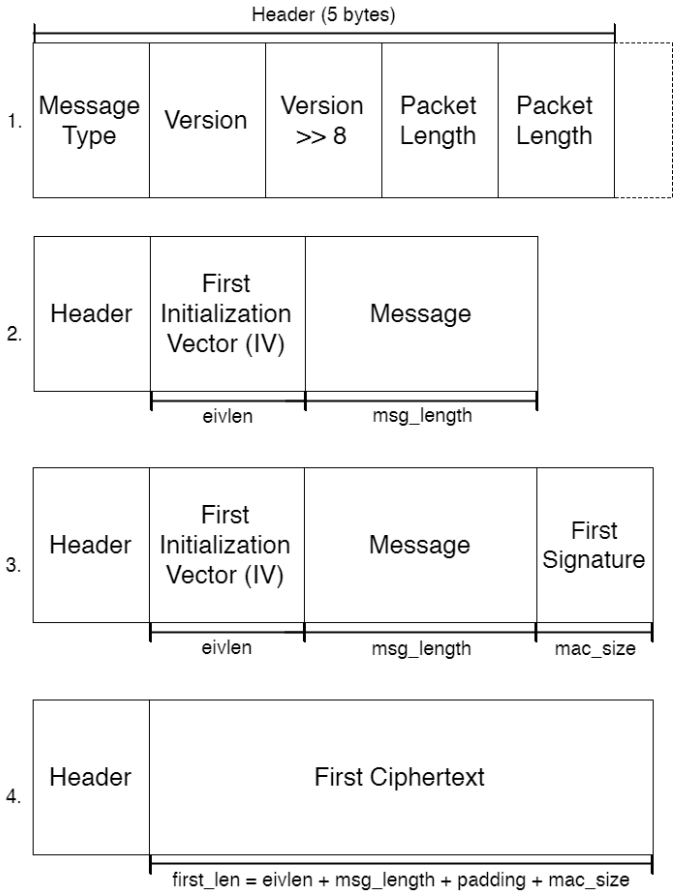

Figure 2: First four steps regarding the ordering of the encryption and signing of SuperTLS using a diversity factor $k = 2$.

3.3.1 Theoretical message size increase

Diversity and redundancy are the essential concepts of SuperTLS and both concepts are used in order to aim increasing security. Nevertheless, signing and encrypting k times has its cost, namely regarding the message size. Figures 2 and 3 show the theoretical increase of the message size originated by using a second MAC and a second encryption function. Considering $eivlen$ the size of the initialization vector (IV) associated with first cipher, msg_length the original message size and mac_size the size of the first MAC, Figure 2 shows that the size of the first ciphertext, $first\_len$, using just one encryption and one MAC is equal to $eivlen + msg\_length + mac\_size$. If the encryption function requires a fixed block size, an additional padding of size $padding$ is also added, making the final size $first\_len = eivlen + msg\_length + padding + mac\_size$. This is the expected size of an OpenSSL message.

SuperTLS, as Figure 3 shows, and as stated before, uses an additional MAC and encryption function. Considering $eivlen$ sec the size of the IV associated with the second cipher

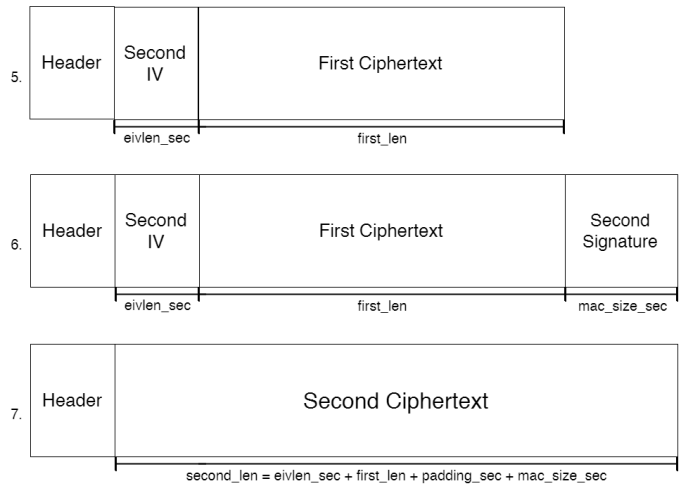

Figure 3: Remaining three steps regarding the ordering of the encryption and signing of SuperTLS using a diversity factor $k = 2$. Here is represented the second signing and the second encryption, employing diversity and redundancy in the communication.

and *mac_size_sec* the size of the second MAC, the size of the second ciphertext is equal to $eivlen\_sec + first\_len + mac\_size\_sec$. Although, similar to the first cipher, if the encryption function requires a fixed block size, a padding of size *padding_sec* is also added. Concluding, the final size of the message to be sent is, excluding the header, $eivlen\_sec + first\_len + padding\_sec + mac\_size\_sec$.

In the best case, the number of packets is the same for OpenSSL and SuperTLS. In the worst case, there is sent one additional packet if the encryption function requires a fixed block size and the maximum size of the packet, after the second MAC and the second encryption, is exceeded by, at least, one byte. In this case, an additional full packet is needed to be sent due to the constraint of having fixed block size.

Concluding, the increase of the message size depends on the length of the second Initialization Vector, the possibly-existent second padding and the length of the second generated MAC.

4. EXPERIMENTAL EVALUATION

5. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

6. FUTURE WORK

- Usar duas libraries compativeis e usar funcoes duma e doutra
- Using SHA-3 for hash to increase diversity when it is supported by TLS

7. REFERENCES

[1] R. Carvalho. Authentication Security through Diversity and Redundancy for Cloud Computing. Master's thesis, Instituto Superior Técnico, Lisbon, Portugal, 2014.

[2] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2 (RFC 5246), 2008.

[3] V. Gupta, S. Gupta, S. Chang, and D. Stebila. Performance analysis of elliptic curve cryptography for ssl. In *Proceedings of the 1st ACM Workshop on Wireless Security*, WiSE '02, pages 87–94, New York, NY, USA, 2002. ACM.

[4] S. Kent and K. Seo. Security Architecture for the Internet Protocol (RFC 4301), 2005.

[5] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture (RFC 4251), 2006.