IMPLEMENTATION AND SIMULATION OF SECURE SOCKETS LAYER (SSL) IN WINDOWS PRESENTATION FOUNDATION

by

Harsh Vachharajani
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Information Security and Assurance

Committee:

_____________________________________ Dr. Kris Gaj, Thesis Director

_____________________________________ Dr. Jim Jones, Committee Member

_____________________________________ Dr. Robert Simon, Committee Member

_____________________________________ Dr. Sanjeev Setia, Department Chair

_____________________________________ Dr. Kenneth S. Ball, Dean, The Volgenau School of Engineering

Date: _______________________________ Summer Semester 2015
George Mason University
Fairfax, VA

Implementation and Simulation of Secure Sockets Layer (SSL) in Windows Presentation Foundation

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Harsh Vachharajani
Bachelor of Technology
Ganpat University, India, 2013

Director: Kris Gaj, Associate Professor
Department of Electrical and Computer Engineering

Summer Semester 2015
George Mason University
Fairfax, VA

ACKNOWLEDGEMENTS

I would not miss this opportunity to thank many people involved with my successful completion of the thesis, without whom I could not possibly think of having achieved this goal. My biggest gratitude goes to my thesis director Dr. Kris Gaj, under whom I initiated my research work, further extending it into a thesis, who guided and helped me at each and every step and had a great contribution toward my successful completion of the thesis. Drs. Jones and Simon, the other two members of my committee were of invaluable help in my research. I would like to especially thank two of my friends, Vikram Gawade and Tejas Dakve, for their invaluable input and contribution of their selfless help to my research, implementation, and solving any issues when I was stuck. A big thanks also goes to Devin Kim, a student under Dr. Gaj’s supervision, who helped to review some portions of my thesis for improvements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Advanced Encryption Standard	AES
Application Programming Interface	API
Certification Authority	CA
Cipher Block Chaining	CBC
Data Encryption Standard	DES
Digital Signature Algorithm	DSA
Hashed Message Authentication Code	HMAC
Hyper-Text Transfer Protocol Secure	HTTPS
Hyper-Text Transfer Protocol	HTTP
Initialization Vector	IV
Message Authentication Code	MAC
Message Digest	MD
Microsoft Developer Network	MSDN
Secure Hash Algorithm	SHA
Secure Sockets Layer	SSL
Transport Layer Security	TLS
Windows Presentation Foundation	WPF
World Wide Web	WWW

ABSTRACT

IMPLEMENTATION AND SIMULATION OF SECURE SOCKETS LAYER (SSL) IN WINDOWS PRESENTATION FOUNDATION

Harsh Vachharajani, M.S.

George Mason University, 2015

Thesis Director: Dr. Kris Gaj

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols designed to provide secure communication between a web server and a web browser. This thesis introduces a unique approach to understanding SSL and TLS by visualizing these protocols through simulation in WPF (Windows Presentation Foundation). This approach is intended to help students to get a clearer picture of how these protocols actually work by clearly illustrating all steps of SSL and TLS, including the handshake and the record protocols.

CHAPTER 1: INTRODUCTION

In the rapidly growing technology era, WWW (World Wide Web) provides a platform for exchanging data, such as text, images, videos, etc. There is a growing need to protect this data, as the exchanged information might be sensitive or confidential. An attacker might intercept the data and try to manipulate it by modifying it, thus violating confidentiality and/or integrity of the message.

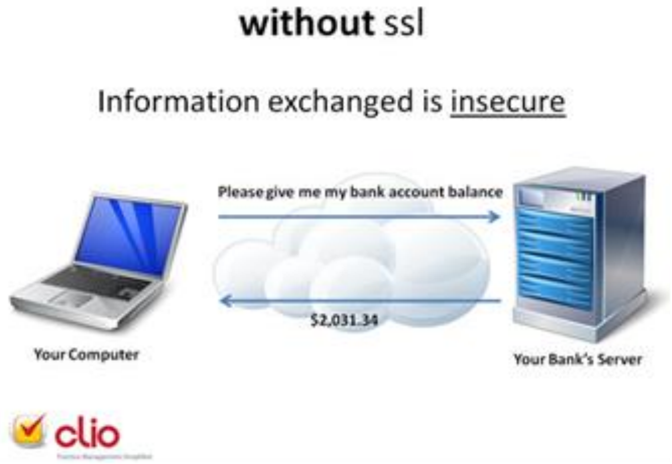

without ssl
Information exchanged is insecure
Please give me my bank account balance
$2,031.34
Your Computer
Your Bank's Server
clio

Figure 1: Without SSL [16]

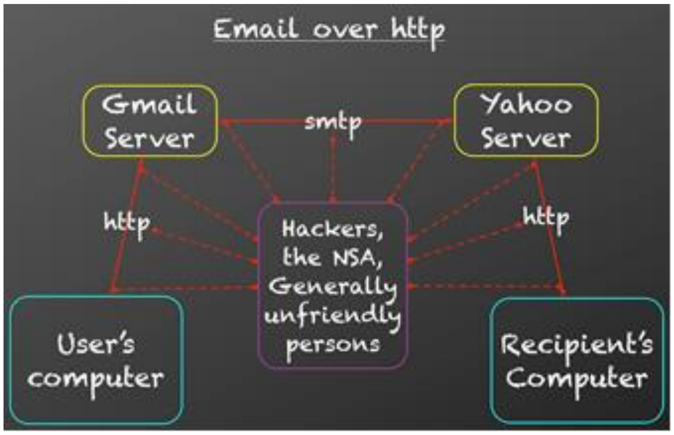


Figure 2: Insecure communication – example [15]

As shown in Fig. 2, a user may be sending a message to the recipient on the opposite side via Gmail, and the recipient may be receiving it using his Yahoo account. Then, the connection between the User and the Gmail server, connection between the Gmail server and the Yahoo server, and the connection between the Yahoo server and the Recipient's computer can be all sniffed by an attacker, such as a hacker, if they are not secured.

In order to protect data being transmitted, we use SSL or TLS, which are cryptographic protocols providing security for the communication over an insecure channel. These protocols establish an encrypted and authenticated connection between a web server and a web browser. With the use of SSL/TLS, an attacker can still intercept the data, however, as the data is encrypted, the attacker is not able to extract any useful information from transmitted packets. Thus, the confidentiality and integrity of the message are preserved.

with ssl
Information exchanged is encrypted for security
01101010001010110101010100101010
11010001110
Your Computer
Your Bank's Server
clio

Figure 3: With SSL [16]

As a part of this thesis, I will try to illustrate the operation of SSL/TLS in a practical manner, visualizing the SSL handshake and the record layer protocols through simulation, with the help of a desktop-based application, developed in C# programming language, using WPF (Windows Presentation Foundation), working in Visual Studio 2013. To enhance this application’s functionality, I have made use of inbuilt MSDN cryptographic library along with the external SSL library named Mentalis Security Library, also developed in C#, which explicitly provides various security related functions, grouped into SecureSocket Library, CertificateServices Library and Crypto Library.

CHAPTER 2: PREVIOUS WORK

There exist several open-source programs, which can be used for teaching cryptography. The most important ones include CrypTool, MAGMA, SageMath, and GnuPG (Gnu Privacy Guard).

2.1 CrypTool

CrypTool is an open-source e-learning software, available for free. The system allows users to experiment with various cryptographic algorithms. The CrypTool project develops the world most-widespread free e-learning programs in the area of cryptography and cryptanalysis.

The CrypTool Portal is devoted to several different versions of software, such as CrypTool 1 (CT1), CrypTool 2 (CT2), JCrypTool (JCT), and CrypTool Online (CTO).

2.1.1 CrypTool 1

CrypTool 1 (CT1) was the first version of CrypTool. It was released in 1998 and allows to experiment with various cryptographic algorithms. CT1 is written in C++ and runs under Windows OS.

It is available in 5 languages, and is the most wide-spread e-learning software of its kind. It can be used for educational purposes in schools and universities.

In particular, CrypTool 1 includes implementations of:

- Numerous classical and modern cryptographic algorithms (encryption, decryption, and key generation)
- Visualization of several algorithms and cryptographic schemes (Caesar, Enigma, RSA, Diffie-Hellman, digital signatures, AES, etc.)
- Cryptanalysis of several algorithms (e.g., Vigenère and RSA with small moduli sizes)
- Cryptanalytical methods (e.g., calculations of entropy, n-grams, autocorrelation, etc.)
- Related auxiliary methods (primality tests, factorization, base64 encoding, etc.)

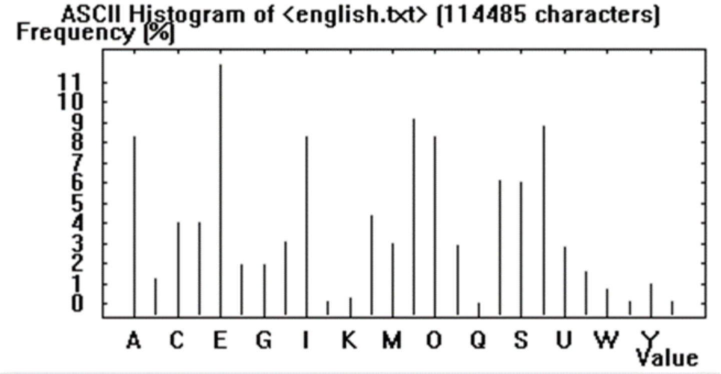


Figure 4: CrypTool 1: Example of a histogram representing relative frequencies of letters in English language.

2.1.2 CrypTool 2 (CT2)

CrypTool 2 is the modern successor to CrypTool 1 and supports visual programming and execution of cascades of cryptographic operations. CT2 also runs under Windows.

CT2 provides various functionalities such as,

Modern Plug'n'Play Interface / Visual Programming - CrypTool 2 provides a graphical user interface for visual programming, and its workflows can be visualized and controlled to enable intuitive manipulation and interaction among various cryptographic functions. The vector-oriented GUI is based on the Windows Presentation Foundation (WPF), which enables users to visualize the functionality of a particular module and understand it in a better way. CrypTool 2 has been developed using C# language. It is based on .NET Framework, and has a pure-plugin architecture, which makes it very easy to develop new additional functions.

Comprehensive Cryptanalysis Functions – CrypTool 2 provides a variety of cryptanalytical tools to analyze or even break classical and modern ciphers. With the current version one can, for instance, apply a ciphertext-only attack on an Enigma-encrypted ciphertext. Plenty of other cryptanalysis functions are also available, such as performing frequency tests to find n-grams and assessing the length of a codeword used for a polyalphabetic cipher.

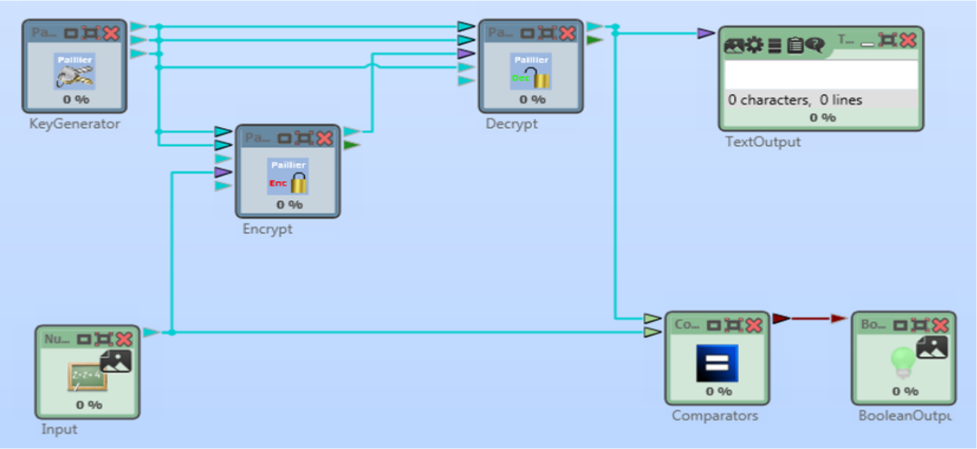


Figure 5: CrypTool 2: Example of encryption, decryption, and comparison between the original and decrypted message.

CT2 also is an open-source project, licensed under Apache Open Source License 2.0, and anyone is welcome to contribute to this project.

2.1.3 JCrypTool

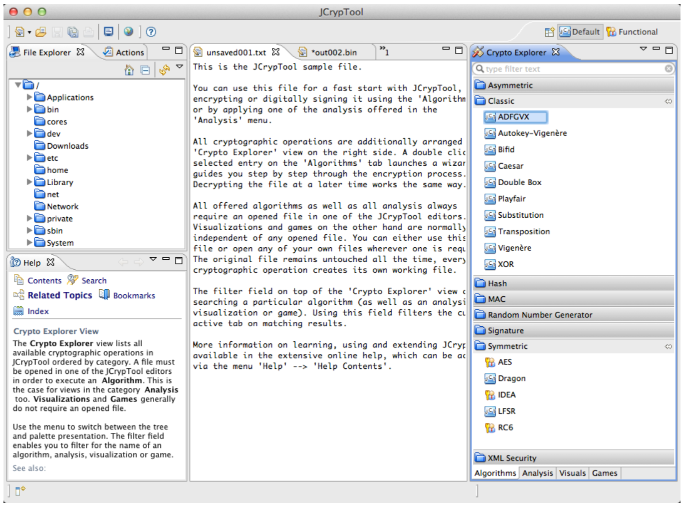
JCrypTool (JCT) is an open-source e-learning platform that allows users to experiment comprehensively with cryptography on Linux, MAC OS X, and Windows. The program enables analysing cryptographic algorithms in a modern, easy-to-use application. JCT platform creates a new way for the users to develop their own cryptographic plug-ins and extend the JCrypTool platform in new directions.

JCT is written in Java and has an Eclipse Rich Client Platform. It includes plenty of cryptographic mechanisms, such as classical, symmetric, and asymmetric ciphers, hash functions, analysis tools, visualizations, and crypto games.

JCT is divided into two parts: JCrypTool Core and JCrypTool Crypto.

The JCrypTool Core Project takes care of the JCrypTool platform, which contains the bare runtime functions, editors, and crypto procedures, as well as core functionality, such as the Crypto Explorer view or the Actions view.

The JCrypTool Crypto Project's aim is to develop new crypto plug-ins (algorithms, analysis, games, visualizations and others) and integrate them into the JCrypTool.



```
This is the JCrypTool sample file.

You can use this file for a fast start with JCrypTool,
encrypting or digitally signing it using the 'Algorithm
or by applying one of the analysis offered in the
'Analysis' menu.

All cryptographic operations are additionally arranged
'Crypto Explorer' view on the right side. A double clic
selected entry on the 'Algorithms' tab launches a wizar
guides you step by step through the encryption process.
Decrypting the file at a later time works the same way.

All offered algorithms as well as all analysis always
require an opened file in one of the JCrypTool editors.
Visualizations and games on the other hand are normally
independent of any opened file. You can either use this
file or open any of your own files wherever one is requ
The original file remains untouched all the time, every
cryptographic operation creates its own working file.

The filter field on top of the 'Crypto Explorer' view c
searching a particular algorithm (as well as an analysi
visualization or game). Using this field filters the cu
active tab on matching results.

More information on learning, using and extending JCryp
available in the extensive online help, which can be ac
via the menu 'Help' --> 'Help Contents'.
```

Crypto Explorer View

The **Crypto Explorer** view lists all available cryptographic operations in JCrypTool ordered by category. A file must be opened in one of the JCrypTool editors in order to execute an **Algorithm**. This is the case for views in the category **Analysis** too. **Visualizations** and **Games** generally do not require an opened file.

Use the menu to switch between the tree and palette presentation. The filter field enables you to filter for the name of an algorithm, analysis, visualization or game.

See also:

Figure 6: JCrypTool GUI.

2.1.4 CrypTool Online (CTO)

CrypTool Online, released in 2009, is the online version of the free e-learning program CrypTool. It allows users to try out different algorithms in a browser or on a smartphone. It provides a huge variety of encryption methods and analysis tools, including many illustrated by examples.

It encrypts directly within a browser and provides an exciting insight into the world of cryptology. CrypTool Online is primarily intended for studying the fundamentals of classical ciphers. The download version of CrypTool is also suitable for working with longer texts and conducting high performance analyses on encrypted messages.

Additionally, CTO includes a Password generator which provides a user with a safe key. The user can also learn about what a “good” password is and how to remember a seemingly complicated password without a problem.

We looked at different open source projects in the area of cryptology. The software which matches most closely the work being done in this thesis is CrypTool. Comparing different versions of CrypTool, CT2 is the best version in terms of visualization of algorithms through simulation, where a user can understand the algorithm / protocol in a much better way, which is the main goal of any educational software. However neither SSL nor TLS protocols have been incorporated in this tool. Hence, our initial idea was to develop their implementations in CT2. Unfortunately, due to the way CT2 operates, this task appeared to be impossible to accomplish, without a substantial redesign of the program. As a result, I decided to develop an educational program devoted to SSL/TLS as a standalone application, which would serve the same purpose as CrypTool, in order to maintain the goal of demonstrating / learning the protocol in an interactive matter. This application was developed with the help of the same development tools, namely, the Windows Presentation Foundation (WPF).

2.2 MAGMA

Magma (also known as Matrix Algebra on GPU and Multicore Architectures) is a freely available software, designed for computations in algebra, number theory, algebraic geometry, and algebraic combinatorics. Magma provides an environment for working with structures such as groups, rings, fields, modules, algebras, schemes, curves, graphs, designs, codes and many others. The main feature is to be able to construct canonical representations of structures, hence making operations such as membership testing, the determination of structural properties, and isomorphism testing, possible.

Magma contains implementation of structures in five branches of algebra: group theory, ring theory, field theory, module theory, and the theory of algebras.

The main features of the Magma include:

- Algebraic Design Philosophy
- Universality
- Integration
- Performance.

Magma can be used to easily encode all public key operations used in SSL and TLS, but it lacks support for secret-key cryptography.

2.3 SageMath

SageMath is a free open-source mathematics software system, which supports research and teaching in algebra, geometry, number theory, cryptography, numerical computation, and related areas. The system is licensed under the GPL. It builds on top of many existing open-source packages: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R, and many more. SageMath uses the Python programming language, supporting procedural, functional, and object-oriented constructs.

The main purpose of SageMath was to create a viable open-source alternative to Magma, Maple, Mathematica, and Matlab.

Some of the features of SageMath include:

- A text-based command-line interface using IPython
- Support for parallel processing using multi-core processors, multiple processors, or distributed computing
- Libraries of elementary and special mathematical functions
- Libraries of number theory functions
- Support for complex numbers, arbitrary precision, and symbolic computation.

Sage integrates many specialized mathematics software packages into a common interface, for which a user needs to know only Python. However, Sage contains hundreds of thousands of unique lines of code, adding new functions, and creating the interface between its components.

Both binaries and source code for SageMath are available from the download page [30, 31]. SageMath is available for multiple operating systems, including Windows, Linux, OS X, and Solaris.

Similarly to Magma, SageMath can be used to easily encode all public key operations used in SSL and TLS, but it lacks support for secret-key cryptography.

2.4 GnuPG

GnuPG (also known as Gnu Privacy Guard) is a free software replacement for the PGP suite of Symantec software. It is a complete and free implementation of the OpenPGP standard, designed to interoperate with PGP; the e-mail encryption program.

GnuPG is a hybrid encryption software program in that it uses a combination of conventional symmetric-key cryptography for speed, and public-key cryptography for ease of secure key exchange, typically by using the recipient's public key to encrypt a session key, which is used only once.

There are different versions of GnuPG. GnuPG 2.x series use Libgcrypt as an encryption library, while GnuPG 1.x series use an integrated library.

GnuPG encrypts messages using asymmetric key pairs, individually generated by GnuPG users. Public keys may be exchanged with other users in various ways, e.g., using Internet key servers. It is also possible to add a cryptographic digital signature to a message, so the message integrity and sender identity can be verified.

GnuPG also supports symmetric encryption algorithms.

At the time of writing this thesis, GnuPG supports the following algorithms:

- Public key: RSA, ElGamal, DSA
- Secret key: IDEA, 3DES, CAST5, Blowfish, AES-128, AES-192, AES-256, Twofish, Camellia-128, Camellia-192, Camellia-256
- Hash: MD5, SHA-1, RIPEMD-160, SHA-256, SHA-384, SHA-512, SHA-224
- Compression: ZIP, ZLIB, BZIP2.

Some of the applications that support GPG includes:

Claws mail – an email client with GPG plugin

GPG4win - a Windows package with tools and manuals for email and file encryption

GPGMail – an OS X Mail.app plug-in

GPGTools – an OS X package with tools for email and file encryption (GPGMail, GPG Keychain Access, MacGPG2, GPG Services, etc.).

The source codes of GnuPG could be used as a basis for our program, however using them would require a substantial effort devoted to rewriting some functions (as the OpenPGP protocol is different than SSL/TLS) and mixing code written in different programming languages.

CHAPTER 3: BACKGROUND OF SSL/TLS AND ATTACKS ON IT

SSL (Secure Sockets Layer) is a cryptographic protocol designed to provide secure communication over a computer network. It is a standard technology that establishes an encrypted link between a server and a client – typically a web server and a browser. HTTPS (also called HTTP over TLS) is HTTP (Hypertext Transfer Protocol) protected by SSL or TLS.

The SSL protocol includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format used to exchange data. The SSL handshake protocol involves a series of messages being exchanged between an SSL-enabled server and an SSL-enabled client, when they first establish an SSL connection, which happens before the SSL record protocol.

This exchange of messages serves the following purposes:

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets.
- Establish an encrypted SSL connection.

The SSL protocol supports a variety of different cryptographic algorithms. It allows various algorithms to be used in operations, such as, authenticating the server and client to each other, exchanging certificates, and establishing session keys. Clients and servers may support different sets of cipher suites, depending on the SSL version they support. Among its other functions, the SSL handshake protocol determines how the server and client will decide on which cipher suites they will use to authenticate each other, to exchange certificates, and to establish session keys.

The protocol combines the following three points to provide communication security:

Privacy - connection through encryption,

Identity authentication – identification through certificates,

Reliability – dependable maintenance of a secure connection through message integrity checking.

3.1 Differences between SSL and TLS

SSL was developed by Netscape. The first version, named SSL v1.0, was not released and version 2.0 had a number of security flaws, which led to the release of SSL v3.0. The differences between versions 2.0 and 3.0 are summarized in Table 1.

Table 1: Difference between SSL version 2 and SSL version 3

	SSL version 2	SSL version 3
Security Improvements	Vulnerable to a "man-in-the-middle" attack. An active attacker can invisibly edit the list of	Defends against this attack by having the last handshake message include a hash of all the previous handshake

	SSL version 2	SSL version 3
	ciphersuite preferences in the “**hello messages**” to invisibly force both client and server to use 40-bit encryption.	messages.
	Uses a weak MAC construction	Uses a strong MAC construction.
	Feeds padding bytes into the MAC in block cipher modes, but leaves the padding-length field unauthenticated, which could allow active attackers to delete bytes from the end of messages.	Fixed the issue of leaving the padding-length field unauthenticated.
	Message Authentication uses only 40 bits when using an Export cipher.	The Message Authentication Hash uses a full 128 bits of keying material, even when using an Export cipher.
Functionality Improvements	The client can only initiate a handshake at the beginning of the connection.	The client can initiate a handshake routine even in the middle of an open session. A server can request that the client starts a new handshake. Thus, the parties can change the algorithms and keys used whenever they want.
	Does not allow the server and client to send chains of certificates.	Allows the server and client to send chains of certificates. This allows organizations to use a certificate hierarchy that is more than two certifications deep.
	Does not allow for record compression and decompression.	Allows for record compression and decompression
Backward Compatibility	SSL 3.0 can recognize an SSL 2.0 “client hello” and fall back to SSL 2.0. An SSL 3.0 client can also generate an SSL 2.0 “client hello” with the version set to SSL 3.0, so SSL 3.0 servers will continue the handshake in SSL 3.0, and SSL 2.0 server will cause the client to fall back to SSL 2.0.	
Other	SSL 3.0 separates the transport of data from the message layer. In 2.0, each packet contained only one handshake message. In 3.0, a record may contain part of a message, a whole message, or several messages. This requires different logic to process packets into handshake messages. Therefore, the formatting of the packets had to be completely changed.	
	Cipher specifications, handshake messages, and other constants are different.	

3.1.1 TLS 1.0

This TLS protocol was first defined in January 1999 as an upgrade to SSL v3.0. Some of the major differences are:

- Key derivation functions are different
- MACs are different - SSL 3.0 uses a modification of an early HMAC, while TLS 1.0 uses HMAC.
- The Finished messages are different
- TLS has more alerts
- TLS requires DSS/DH support.

3.1.2 TLS 1.1

This version of TLS was defined in April 2006 as an upgrade to TLS v1.0. Some of the major differences are:

- The Implicit Initialization Vector (IV) is replaced with an explicit IV to protect against Cipher Block Chaining (CBC) attacks.
- Handling of padded errors is changed to use the bad_record_mac alert rather than the decryption_failed alert to protect against CBC attacks.
- IANA registries are defined for protocol parameters
- No longer prematurely closes to cause a session to be non-resumable.

3.1.3 TLS 1.2

This TLS version was defined in August 2008. It contained improved flexibility compared to TLS v1.1. Some of the major differences are:

- The MD5/SHA-1 combination in the pseudorandom function (PRF) is replaced with cipher-suite-specified PRFs.
- The MD5/SHA-1 combination in the digitally-signed element is replaced with a single hash. Signed elements include a field explicitly specifying the hash algorithm used.
- Substantial cleanup of the client's and server's ability to specify which hash and signature algorithms they will accept.
- Addition of support for authenticated encryption with additional data modes.
- TLS Extensions definition and AES Cipher Suites were merged in.
- Tighter checking of EncryptedPreMasterSecret version numbers.
- Verify_data length depends on the cipher suite.
- Description of Bleichenbacher/Dlima attack defenses cleaned up.

With attacks, such as POODLE [20], SSL v3.0 is no longer considered as secure to use. Organizations that still use SSL v3.0 for securing their websites are at a great risk of major attacks.

Subsequent versions of TLS — v1.1 and v1.2 are significantly more secure and fix many vulnerabilities present in SSL v3.0 and TLS v1.0. For example, the BEAST attack can completely break web sites running on SSL v3.0 and TLS v1.0 protocols. The newer TLS versions prevent the BEAST, and provide many stronger ciphers and encryption methods.

Taking setting up an e-mail application as an example, different options, such as “no encryption,” “SSL,” and “TLS”, indicate how secure the connection to be initiated will be.

3.2 Different public-key exchange methods supported in SSL/TLS

The key exchange method defines how the shared key used for application data transfer will be agreed upon by the client and the server. SSL 2.0 uses RSA key exchange only, while SSL 3.0 and all subsequent TLS versions support the choice of a key exchange scheme, including variants of the RSA key exchange and the Diffie-Hellman key exchange. We review these methods shortly in the following subsections.

3.2.1 RSA

The secret key that is to be used for encryption of messages in the Record Layer is generated at random by a sender, and encrypted with the receiver’s RSA public key. The public key certificate for the server and, optionally, for the client include the RSA generated public keys.

3.2.2 Fixed Diffie-Hellman (DH)

In this key-exchange method, the public-key certificate contains the Diffie-Hellman public parameters signed by the certificate authority (CA). The client provides its Diffie-Hellman public-key parameters either in a certificate, if client authentication is required, or in a key exchange message. This method results in a fixed secret key between two peers based on the Diffie-Hellman calculation using the fixed public keys.

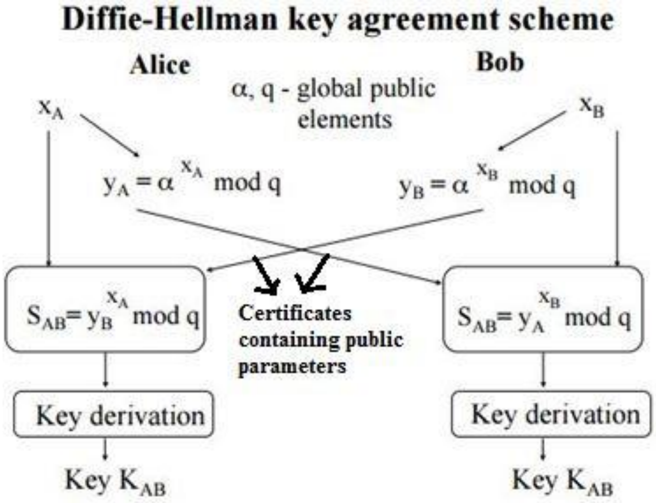

Diffie-Hellman key agreement scheme
Alice
Bob
α, q - global public elements
x_A
x_B
$y_A = \alpha^{x_A} \bmod q$
$y_B = \alpha^{x_B} \bmod q$
$S_{AB} = y_B^{x_A} \bmod q$
Certificates containing public parameters
$S_{AB} = y_A^{x_B} \bmod q$
Key derivation
Key derivation
Key K_{AB}
Key K_{AB}

Figure 7: Fixed Diffie-Hellman

3.2.3 Ephemeral Diffie-Hellman (EDH)

This method is used to create temporary secret keys. Each instance of the protocol uses a different key, hence the same key is never used twice. In this method, the DH public keys are exchanged, signed using the sender’s private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys. This method is the most secure method as it results in a temporary, authenticated key.

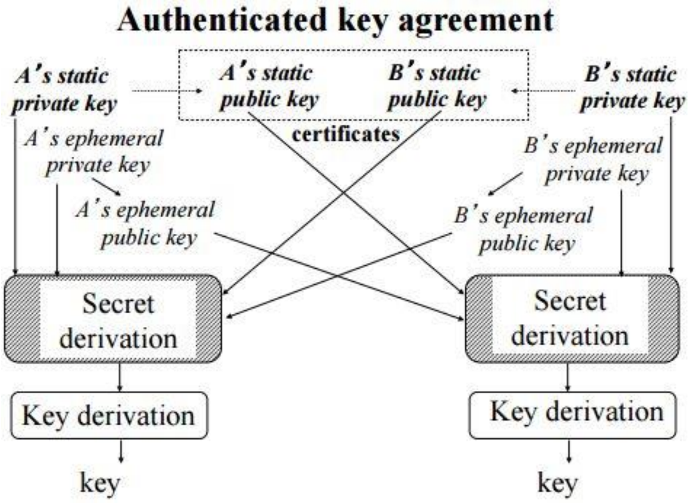


Figure 8: Ephemeral Diffie-Hellman

3.2.4 Anonymous Diffie-Hellman

This method does not use any kind of authentication. Each side sends its public Diffie-Hellman parameters to the other with no authentication. This method is not secure as it can be vulnerable to man-in-the-middle attacks.

3.3 Use of MAC and HMAC

There are different MAC schemes that are used for SSL/TLS. MACs are used to provide data integrity during the transmission of data in the Record Layer. MAC takes the secret key and the arbitrary-length message to be authenticated as the input, and outputs a MAC of a fixed size length. It protects both message integrity as well as its authenticity.

A keyed-hash message authentication code (HMAC) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. Any cryptographic hash function, such as MD5 (128-bits) or SHA-1 (160-bits), may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA1, accordingly. The cryptographic strength of the HMAC depends on the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key.

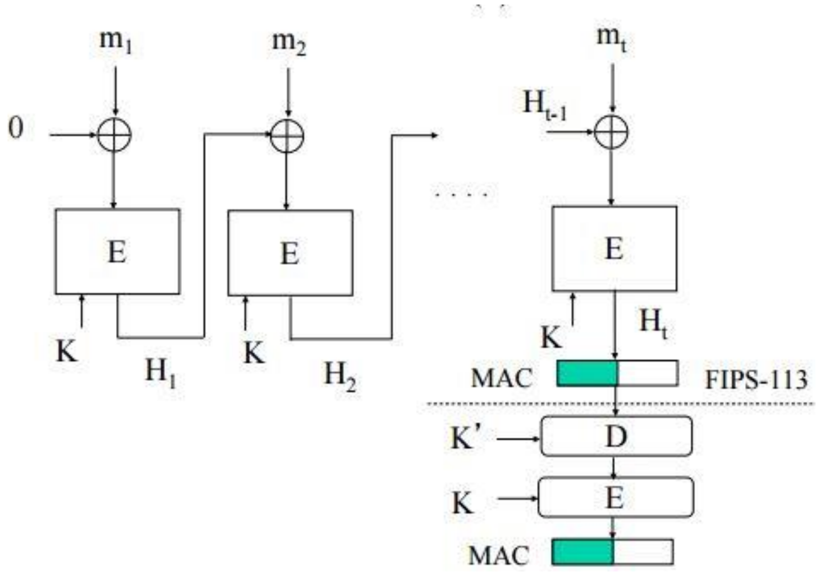


Figure 9: CBC-MAC

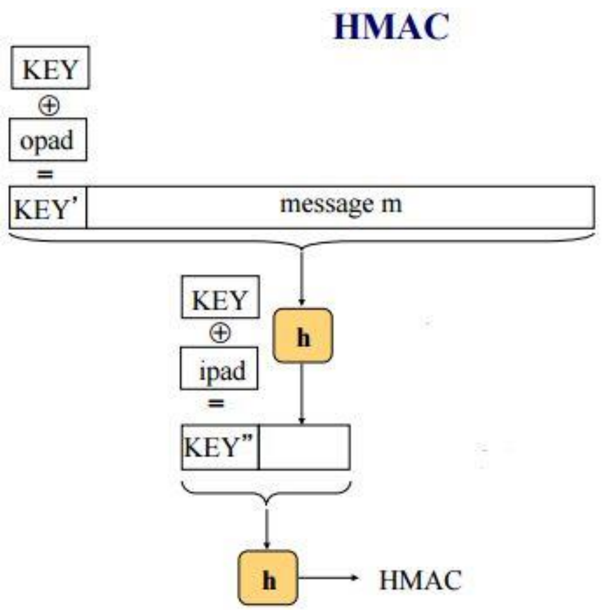


Figure 10: HMAC

3.4 Use of Symmetric cipher suites

Symmetric cipher suites are used to encrypt/decrypt data transmitted in the Record Layer phase, using symmetric-key algorithms selected by the server and the client in the Handshake phase.

Some of the symmetric-key algorithms used are listed below:

DES – Data Encryption Standard (Key size – 56 bits)

3DES – DES used 3 times to ensure more security than DES.

AES – Advanced Encryption Standard (key size – 128 / 192 / 256 bits)

RC2-40 (key size – 40 bits)

3.5 SSL Handshake

An SSL session always begins with an exchange of data using the SSL handshake. The handshake allows the server to authenticate itself to the client using public-key techniques.

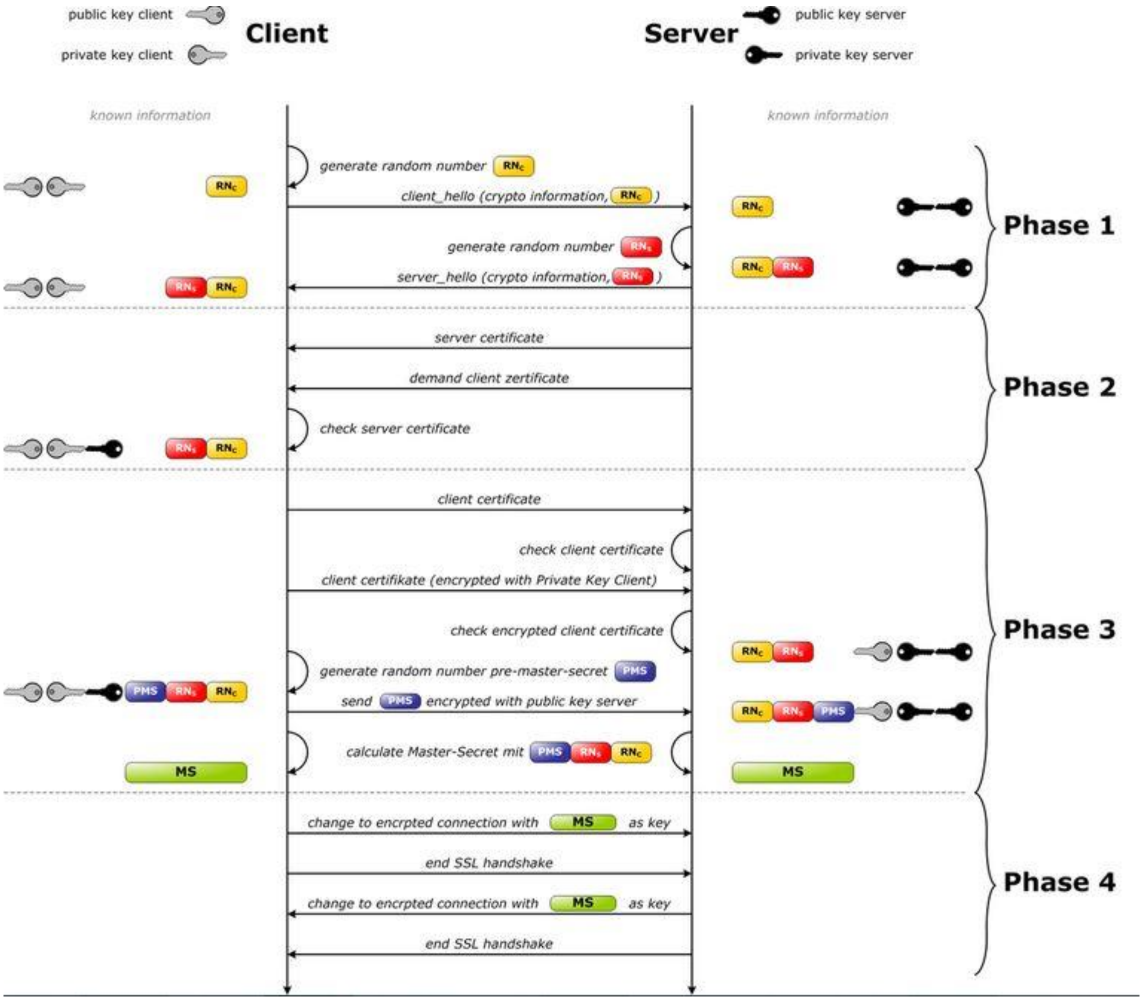

public key client
private key client
Client
Server
public key server
private key server
known information
known information
generate random number RNc
client_hello (crypto information, RNc)
RNc
RNc
generate random number RNs
server_hello (crypto information, RNs)
RNs RNc
RNc RNs
Phase 1
server certificate
demand client zertificate
check server certificate
RNs RNc
Phase 2
client certificate
check client certificate
client certifikate (encrypted with Private Key Client)
check encrypted client certificate
RNc RNs
generate random number pre-master-secret PMS
send PMS encrypted with public key server
PMS RNs RNc
RNc RNs PMS
calculate Master-Secret mit PMS RNs RNc
MS
MS
Phase 3
change to encrpted connection with MS as key
end SSL handshake
change to encrpted connection with MS as key
end SSL handshake
Phase 4

Figure 11: SSL Handshake steps [13]

- The client sends the SSL version number, the cipher suites supported, a random number, and other information needed to communicate to the server.
- The server also sends the SSL version number, the strongest cipher suites chosen and supported by it, a random number, and other information needed to communicate to the client. The server also sends its own certificate. If the client is requesting a server resource that requires client authentication, the server requests the client's certificate.
- The client uses the certificate sent by the server to authenticate the server. If the server cannot be authenticated, the user is informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client moves on to the next step.
- Using all data generated in the handshake so far, the client creates the premaster secret for the session, encrypts it with the server's public key (which is obtained from the server's public key certificate), and sends the encrypted premaster secret to the server.
- If the server has requested client authentication (which is an optional step in the handshake), the client also signs another piece of data with its private key, which is unique to this handshake and known by both the client and server. In this case, the client sends both the signed data and the client's own certificate to the server along with the encrypted premaster secret.
- If the server has requested client authentication, the server also tries to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the

premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the master secret.

- Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity-that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.
- The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate encrypted message indicating that the client portion of the handshake is finished.
- The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate encrypted message indicating that the server portion of the handshake is finished.

The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt data they send to each other and to validate its integrity.

There are six different keys generated for each side; the client and the server. These keys are:

- Server_write_MAC_secret: The secret key used in MAC operations on data sent by the server.
- Client_write_MAC_secret: The client key used in MAC operations on data sent by the client.

- Server_write_key: The symmetric encryption key used for data encrypted by the server and decrypted by the client.
- Client_write_key: The symmetric encryption key used for data encrypted by the client and decrypted by the server.
- Server_IV: Initialization vector maintained by the server for each key when CBC mode is used.
- Client_IV: Initialization vector maintained by the client for each key when CBC mode is used. The client informs the server that all the future messages from client are encrypted with the generated Client_write key. It then sends a separate encrypted message indicating that the client portion of the handshake is finished.

The server sends a message to client informing that future messages from server are encrypted with the generated Server_write key. It then sends a separate encrypted message indicating that the server portion of the handshake is finished.

The client sends a message to the server informing that future messages from client are encrypted with the generated Client_write key. It then sends a separate encrypted message indicating that the client portion of the handshake is finished.

3.6 SSL Record Layer

Communication between the client and the server takes place with the help of SSL Record Protocol. The SSL Record Protocol provides three services for SSL connections: confidentiality, by encrypting application data; and message integrity and authentication, by using MAC.

The following figure illustrates the overall operation of the SSL Record Protocol. The Record Protocol takes the application data to be transmitted, fragments the data into blocks, optionally compresses the data, adds MAC to it, encrypts, appends SSL Record Header, and transmits the resulting unit. On the receiver's side, the received data gets decrypted, verified by computing MAC, decompressed, reassembled, and then gets delivered to the calling application on receiver's side.

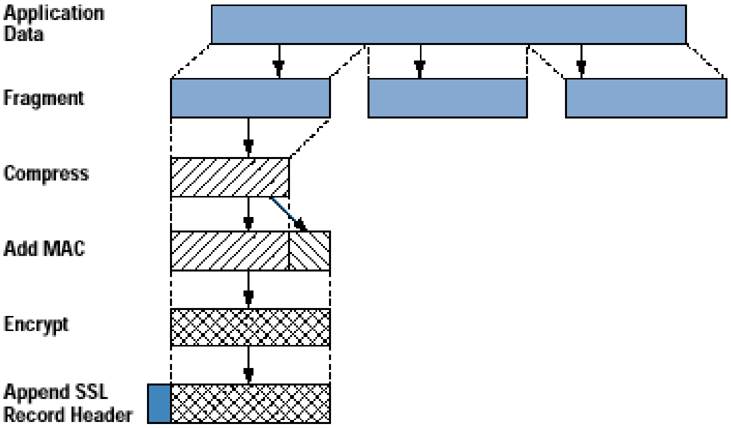


Figure 12: SSL Record Protocol [8]

The first step is fragmentation. Every application data gets fragmented into fixed sized blocks. Next, compression is optionally applied. No compression algorithm is specified in SSL version 3.0, so the default compression algorithm is null. However, depending on the implementation, Record Protocol may include a compression algorithm.

In the next step, a message authentication code gets computed over a compressed data. A hash code gets calculated by a combination of the compressed data, a secret key, and some padding. For this purpose, the Client_write_MAC_secret and Server_write_MAC_secret keys are used. The receiver performs the same calculation and compares the computed MAC value with the incoming MAC value. If the two values match, the receiver is assured that the message has not been altered in transit. Also, by applying MAC, the receiver will have the certainty about the authenticity of the sender.

Next, the compressed data and attached MAC are encrypted using a symmetric key cipher. Finally, the SSL Record Protocol prepends an SSL Record Header and transmits the entire resulting unit to the receiver.

3.7 Attacks on SSL / TLS

We will now have a look at several types of attacks that have been developed against SSL/TLS:

3.7.1 Cipher Suite Rollback Attack

This attack aims at limiting the offered cipher-suite list provided by the client to a weaker one or the one composed of NULL-ciphers. A Man-in-the-middle (MITM) attacker may alter the ClientHello message (sent by the initiator of the connection), strip the undesirable cipher-suites or completely replace the cipher-suite list with a weak one, and pass the manipulated message to the desired recipient. The server has no real choice - it can either reject the connection or accept the weaker cipher-suite.

This problem was fixed with the release of SSL 3.0, through authenticating all messages of the Handshake protocol, by including the hash value of all messages sent and received by the client in the ClientFinished message.

3.7.2 Version Rollback Attack

In a version rollback attack, a ClientHello message of SSL 3.0 is modified to look like a ClientHello message of SSL 2.0. This forces the server to switch back to the more vulnerable SSL 2.0. As a prevention, the SSL/TLS version is also contained in the PreMasterSecret of the ClientKeyExchange message.

3.7.3 BEAST

BEAST leverages a type of cryptographic attack called a chosen-plaintext attack. The CBC vulnerability can enable man-in-the-middle (MITM) attacks against SSL in order to silently decrypt and obtain authentication tokens, providing hackers with access to the data passed between a Web server and the Web browser.

The attacker mounts the attack by choosing a guess for the plaintext that is associated with a known ciphertext. To check if a guess is correct, the attacker needs access to an encryption oracle to see if the encryption of the plaintext guess matches the known ciphertext. An attacker observing 2 consecutive ciphertext blocks C0, C1 can test if the plaintext block P1 is equal to x by choosing the next plaintext block $P2 = x \oplus C0 \oplus C1$. Due to the way how CBC works, C2 will be equal to C1 if x = P1.

The vulnerability of the attack had been fixed with TLS 1.1 in 2006, but TLS 1.1 had not seen wide adoption prior to this attack demonstration. As a preventive measure, RC4 is used as it is immune to BEAST attack, hence it was used to mitigate this attack on the server side.

3.7.4 POODLE

This attack, named as Padding Oracle On Downgraded Legacy Encryption, is a vulnerability in the design of SSL 3.0, which makes CBC mode with SSL 3.0 vulnerable to a padding attack. POODLE allows a man-in-the-middle attack, through means of a malicious Wi-Fi hotspot or a compromised ISP to extract data from secure HTTP connections, which could allow the attacker to access online banking and use the victim's session for malicious purposes. The attackers only need to make 256 SSL 3.0 requests to reveal one byte of encrypted messages.

As this vulnerability exists only in SSL v3.0, and currently most of the browsers use TLS 1.0 and above, this attack works only if the attacker can successfully conduct a version rollback attack first. To overcome this attack, the users need to disable SSL v3.0 in their browsers and enable TLS 1.0.

3.7.5 HeartBleed Attack

The Heartbleed bug is a serious vulnerability specific to the implementation of SSL/TLS in the popular OpenSSL cryptographic software library. This weakness allows attackers to steal private keys from servers that should normally be protected. The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This flaw in the code was related to the bad input validation with the data length, meaning that the length of the data sent was not properly validated against the SSL3_RECORD's length field because of which the attacker was successfully able to overrun the memory, thus adding his malicious code to it. This attack compromised the secret and private keys, the names and passwords of the users, and the sensitive information stored on the users computers. The attack allowed attackers to eavesdrop on communication between the browser and the server, steal data directly from services and users, and impersonate services and users. It should be stressed that, this vulnerability was a bug in the OpenSSL software, rather than any error in the SSL or TLS protocol specification.

CHAPTER 4: DEVELOPMENT TOOLS AND LIBRARIES

In terms of the development tools, I have tried to use the same development platform and the programming language as used in CrypTool 2, namely C# as the programming language with .NET Framework 4.0 in Visual Studio 2013, and WPF as a user interface (UI).

C# is an object-oriented language that provides developers a way to build robust and secure applications which run on .NET framework. C# is used to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, etc. C# provides language constructs to directly support these concepts, making it a very natural language in which to create and use software components. C# supports generic methods and types, which provide an improved safety and performance, and iterators, which enable the components that implements the collection classes to define iterative behaviors that are simple to use by client code.

To ensure that C# programs and libraries can evolve over time in a compatible manner, much emphasis has been placed on versioning in C# design. Many programming languages pay little attention to this issue, and, as a result, programs written in those languages break more often than necessary when newer versions of dependent libraries are introduced.

WPF, introduced in .NET framework 3.0 provides a platform for the development of highly functional rich client applications. It is a GUI framework that allows programmers to create an application with a wide range of elements, like labels, textboxes, and other well-known elements. Over the period of time, WPF has become answer for software developers and graphic designers who want to create modern user experiences without having to master several different technologies. As applied to the .NET Framework programming model, XAML simplifies creating a UI for .NET framework applications. There are built-in controls provided as part of WPF, such as buttons, menus, grids, and list boxes.

WPF provides an integrated system for building user interfaces with common media elements, like vector and raster images, audio, and video. WPF also provides an animation system and a 2D/3D rendering system.

The reason WPF was chosen for the implementation was due to its rich composition and customization. As my goal was to develop a simulation program in order to visualize the exchange of messages, WPF provided a variety of options as well as support for XAML markup language. XAML can be used to create custom controls, graphics, 3D images, animations and many more modules that are included.

There are different libraries that provide us with some built-in functionalities that might be useful in implementing our application. In my work too, I have used a few libraries which provide some of the built-in cryptographic functionalities related to SSL/TLS.

4.1 OpenSSL

OpenSSL is a general-purpose cryptography library that provides an open-source implementation of SSL/TLS protocols. It is a robust, full-featured toolkit implementing SSL and TLS.

The core library is written in C programming language and implements basic cryptographic algorithms and schemes. There are different versions available for multiple operating systems, such as Linux, Mac OS X, Solaris, and Windows. The source code for OpenSSL is available from the download page [50].

OpenSSL offers a Command Line Interface (CLI) that is used to generate and manage private and public keys, creation of certificates, calculating message digests, handling of S/MIME signed or encrypted mail.

OpenSSL is used in:

- Generating public and private key pairs
- Generating a Certificate Signing Request (CSR)
- Sending the request to CA and getting the public certificate
- Using the certificate for authentication purposes.

There are different data structures used by the OpenSSL library functions:

- SSL_METHOD
- SSL_CIPHER
- SSL_CTX
- SSL_SESSION
- SSL.

4.2 MSDN Library

MSDN is a set of functions that help developers write applications using various APIs that come along with Microsoft products. This library contains technical documentation and content intended for developers using Microsoft Windows. The library has APIs, source code, technical articles, and other programming information. It can be freely downloaded as a package with Microsoft development tools, like Visual Studio, and can be used when developing applications.

Every version of Visual Studio has an integrated help viewer that helps you to access the then current edition of MSDN library. The library contains different classes, which has constructors, methods, and properties that as a whole provide you with the functionality that the developer is looking for.

Figure 13 describes the tree structure of MSDN library, which defines its hierarchy.

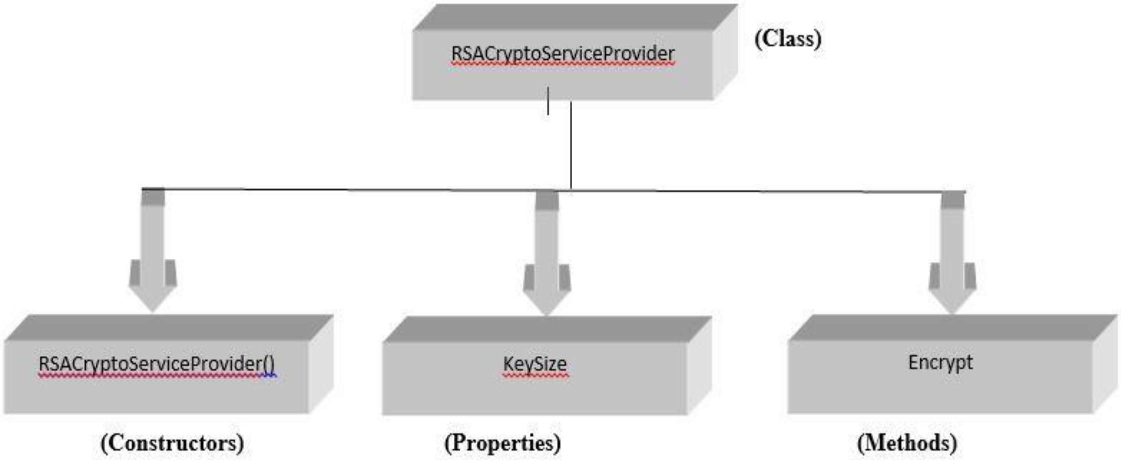


Figure 13 MSDN library hierarchy example

4.3 Mentalis

Mentalis is a free open-source software available online that contains several security libraries called Mentalis.org security library, which is an add-on for .NET framework. The main purpose of this security library is to provide security related functions and tools in C# for VB.NET development.

The library supports the following:

- Authentication
  - A password validation library enables you to check a user password according to set of validation rules.
- Cryptography
  - The algorithms such as HMAC and RC4.
- Smart Cards
  - Offers a framework to connect and communicate with smart cards.

There are several different subprojects that the library consists of:

- SecureSocket Library (SSL and TLS support)
  - This is a free and open source library that implements SSL v3.0 and TLS v1.0. This library contains a SecureSocket class that encrypts and decrypts the data passed through it. It has several other classes that internally implement algorithms and protocols that help in the functioning of SSL/TLS.

- CertificateServices Library (Certificate Management support)
    - This library is an implementation of Certificate API that includes the following features:
        - Methods of X509Certificate class, loading private key files, loading certificates from certificate stores, etc.
        - Building a certificate chain and verifying it
        - Encrypting and decrypting data with respective public and private parameters of the certificates
        - Methods pertaining to creating certificates, revoking certificates, deleting certificates, etc.
- Crypto Library (Cryptographic Transformations support)
    - This library implements several cryptographic algorithms such as RC4, AES, HMAC, etc.
    - It covers Asymmetric Cryptography, Hashing, and Symmetric Cryptography classes and methods.

Figure 14 describes the hierarchy/tree structure of the functionality provided by Mentalis library.

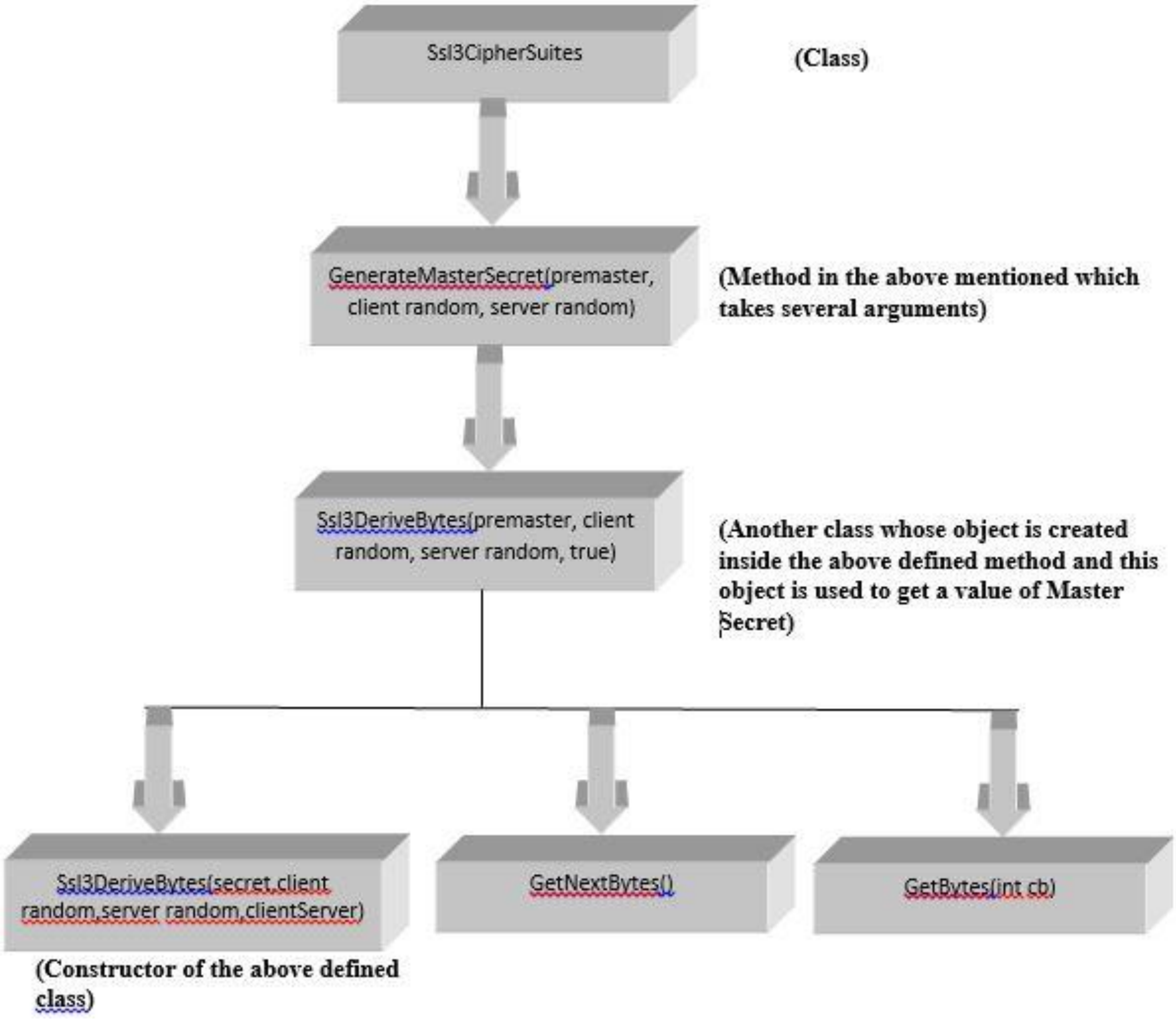

Ssl3CipherSuites
(Class)
GenerateMasterSecret(premaster, client random, server random)
(Method in the above mentioned which takes several arguments)
Ssl3DeriveBytes(premaster, client random, server random, true)
(Another class whose object is created inside the above defined method and this object is used to get a value of Master Secret)
Ssl3DeriveBytes(secret,client random,server random,clientServer)
(Constructor of the above defined class)
GetNextBytes()
GetBytes(int cb)

Figure 14 MSDN library hierarchy example

CHAPTER 5: IMPLEMENTATION

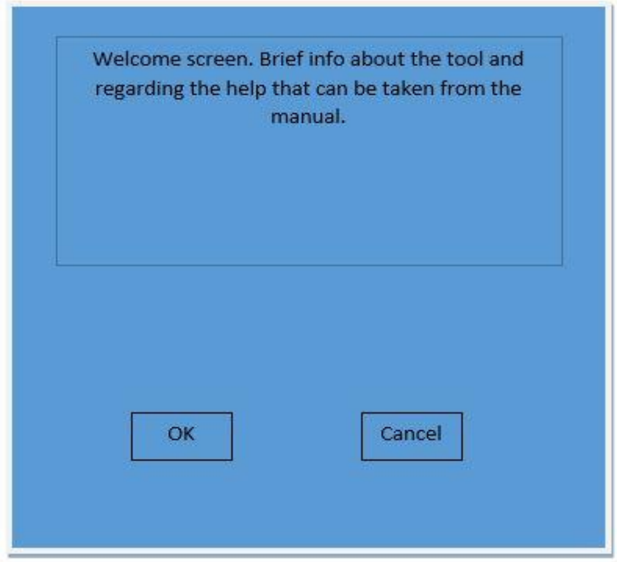
Having described the background of SSL/TLS theoretically in the previous sections, which helps us to understand each and every aspect of SSL/TLS and the functions associated with it, we shall now have a look at the original work, a working model of SSL/TLS, developed as a part of this thesis.

Let us have a look at the implementation and flow of the program.

5.1 The Handshake Phase

5.1.1 Welcome Screen

Right before entering the handshake phase, the tool starts with an introductory welcome screen, which displays a brief introduction about the program and also talks in short about the user manual, from which a user can receive help required to understand the functionality of the program in a detailed manner.



Welcome screen. Brief info about the tool and regarding the help that can be taken from the manual.

Figure 15: Welcome Screen

5.1.2 Generation of Certificate

Moving ahead from the welcome screen, the next part is the certificate generation window where the public key certificate for both, the server and the client is generated. This public key certificate is provided during the handshake phase by the server to the client for authentication purposes, where the client verifies the certificate by using the CA’s public key.

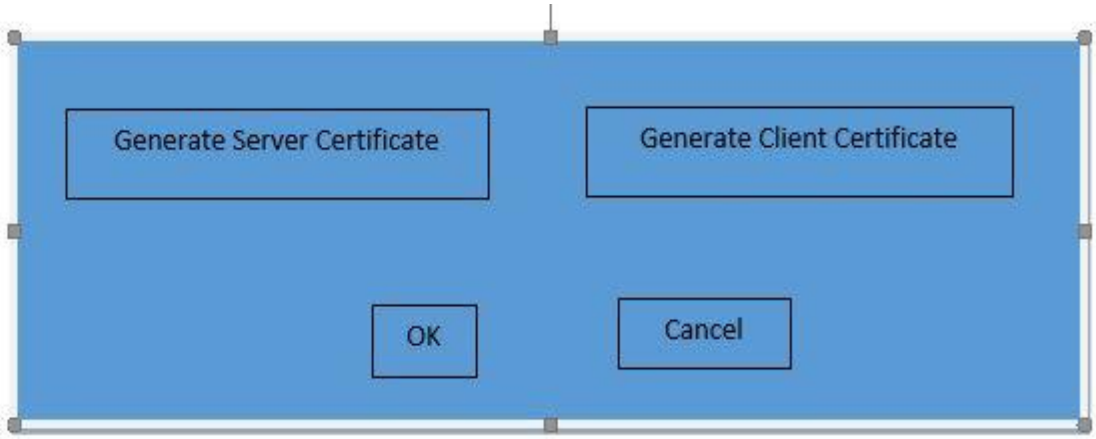


Figure 16: Certificate Generation

As seen in the above image, there are two options provided; one for generating server's public certificate and another for generating the client's certificate (which is optional in real scenario). Clicking the "Generate Server Certificate" button redirects the flow to a window that is shown in Figure 17. As shown in this figure, there is an exchange of messages taking place between the server and the CA. First the server provides his details, which includes the public key of a specific size that the server chooses. These details are then sent over to the CA where the CA will create a public key certificate, signs it with its private key, and sends back to the server. Now the server has his public certificate, which can be used at the handshake phase to authenticate himself to the client.

In this simulation program, it may not be possible to show how the CA actually verifies whether the certificate requesting server is a legitimate/trusted server or not. In reality, the CA generally carries out a two-step validation process:

- Domain name validation: Verifying the authenticity of the domain name owned by the requesting server.
- Verifying whether the server is a legitimate server or not.

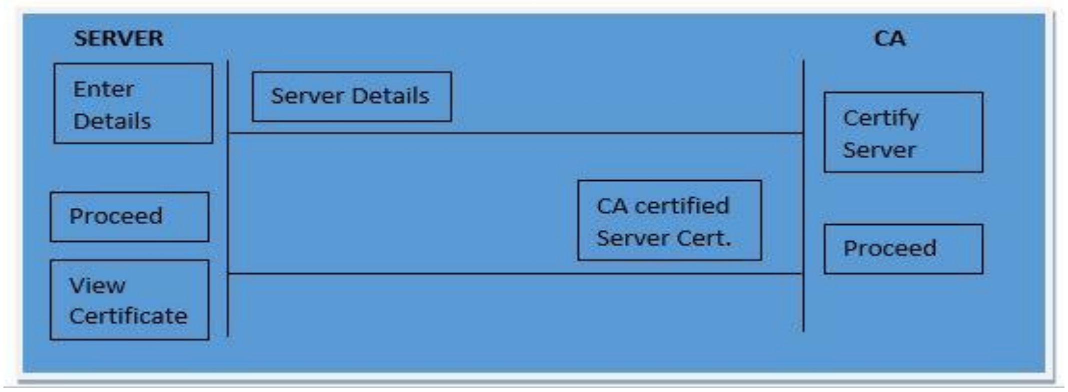


Figure 17: Certificate Generation – Server Details window

In the certificate generation step of the developed program shown in Figure 17, the requesting server is already considered legitimate by the CA because the primary purpose of demonstrating the certificate generation step is just to help the user acquire knowledge about the details of the public-key certificate that will be sent by the server to the client during the handshake phase.

In the similar manner as explained above, the client public certificate is generated, which is usually optional, as normally the client (i.e., the browser in a real scenario) does not need to always authenticate itself to the server.

5.1.3 Mode Selection

After the certificate generation step is done, the flow of the program moves ahead toward the mode selection window, where the user is given an option of selecting one of the two modes – the Demo mode or the Attack mode. Selecting any one of them opens up the corresponding handshake window that demonstrates the inner workings of the handshake protocol.

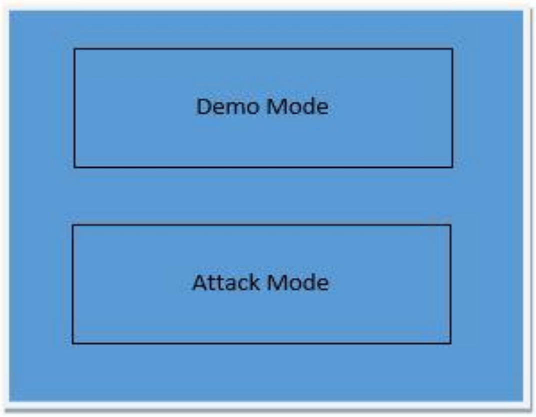


Figure 18: Mode Selection window

5.1.4 The Handshake Protocol

The implementation of the handshake protocol is done in a similar manner to the explanation of this protocol provided in Section 3.5. The image shown below is the actual implementation snapshot of the handshake, which is developed as a part of the thesis.

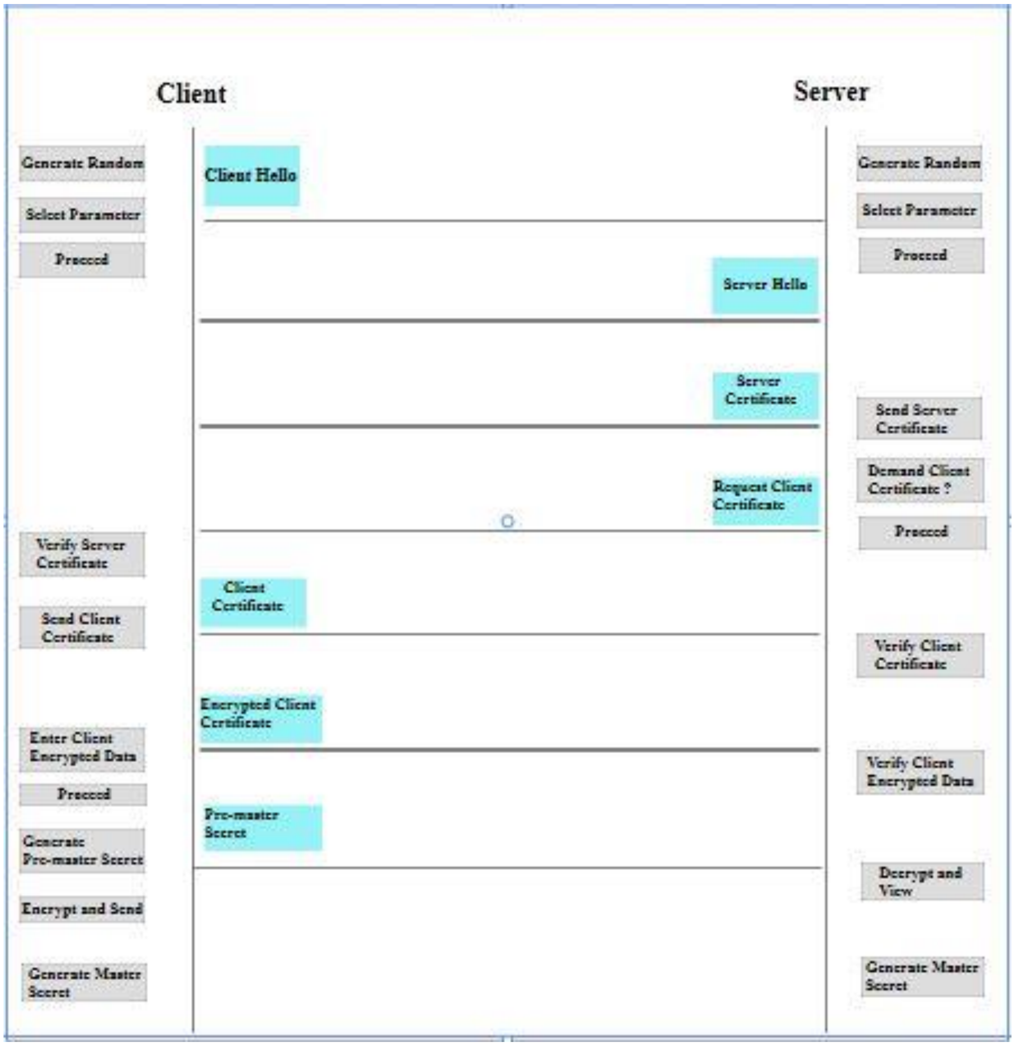


Figure 19: Demo mode Handshake window

The actual working of the program can be seen by clicking on subsequent buttons, shown in Figure 19.

Starting at the client’s side, clicking on “**Generate Random**” button, a 32-byte random number is generated for the client. Moving ahead and clicking on the “**Select Parameters**” button, a window opens up that has cipher suites options to be selected from, such as SSL/TLS version, Asymmetric key algorithms, Symmetric key algorithms and Hashing algorithms. At the end of this step, the “**Client Hello**” message is sent to the server. Then, the user clicks on “**Proceed**” button and the **blue** colored text block with the text **Client Hello** slides across towards the server side, which indicates that the client hello message has been sent over to the server. The sliding of the text block is possible due to the animation feature provided by WPF.

The rest of the program flows in the similar manner, where clicking on a certain button slides a text block to the opposite end, indicating that the message has been sent over to the receiving end.

At the end of the handshake phase, there are six session keys generated that are used for communication between the client and the server in the record protocol. Having obtained the session keys at the end of the handshake phase, the flow of the program moves to the record protocol, where the client-server communication is shown.

5.1.5 The Record Protocol

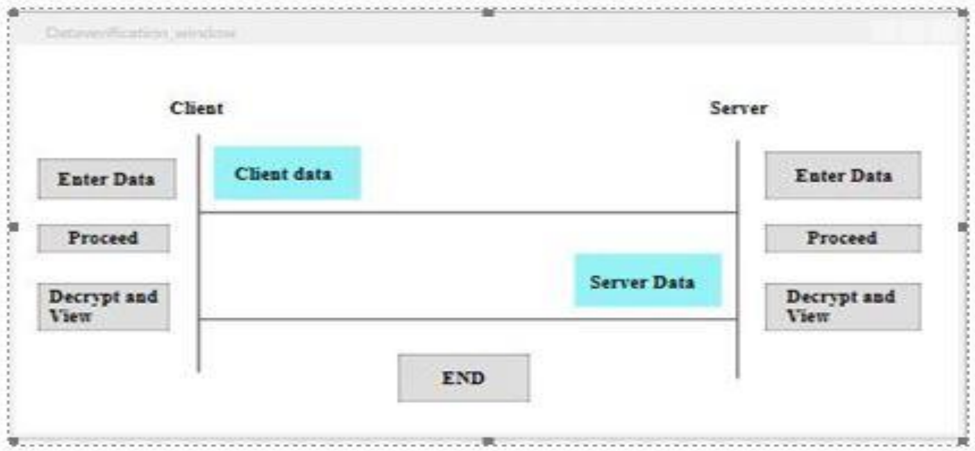


Figure 20: The Record Protocol

Figure 20 is the snapshot of the record protocol window. The client can select any data/file to be sent to the server. Clicking on “Enter Data” button on the client side opens up a new window that allows you to choose a file from your system such as a text file, word file, pdf file, etc. After the selection of the file and before sending it, the file goes through the entire process of **Fragments → Compression → Adding HMAC → Encryption**. It is then sent over to the server where the server does the whole process in a reverse manner (**Decryption → Compare HMAC →Decompression → Assembling the fragments**) to get back the original file.

Considering the implementation of the program, the server follows the same process, as mentioned in the above paragraph. Hence, this process goes on till the client and the server want to communicate.

Talking about the other mode – Attack mode, there are three types of attacks demonstrated, due to which many other attacks are possible.

1. Cipher Suite Rollback Attack
2. Version Rollback attack
3. Man-in-the-middle attack

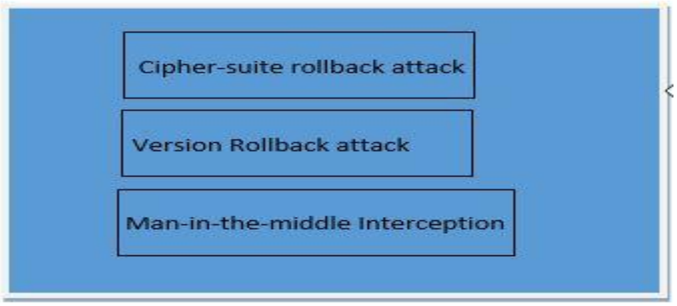


Figure 21: Attack selection window

As seen from the above image, this window opens up by selecting “Attack Mode” from the mode selection window. Selecting any of these attacks directs the program to open the handshake window with additional functionalities available compared to the Demo mode.

Figure 22 displays the Man-in-the-middle attack handshake window, where there is an “Interceptor” button that can be seen. This particular flow of the program is shown from the attacker’s perspective. The attacker as a middle man can intercept the communication that is going on between the client and the server. By clicking on the “Intercept” button, the attacker stops the “client hello” message that is sent by the client and instead drafts its own message and sends it to the server, thus impersonating the client. The server on the other side treats this message as coming from the actual client, and sends his response back to the client. The attacker does the same thing here as well; keeps the server message and generates its own server message, thus impersonating the server.

Hence, with this attack demonstrated, a conclusion can be drawn that all the communication thereafter can be intercepted and decrypted by the attacker, thus creating a security breach and violating confidentiality and integrity services.

Continuing this attack with the record protocol, the attacker can actually modify the content of any message as a middle man, and intercept the whole communication of messages exchanged between the client and the server. In this way, the integrity of the message is no more preserved.

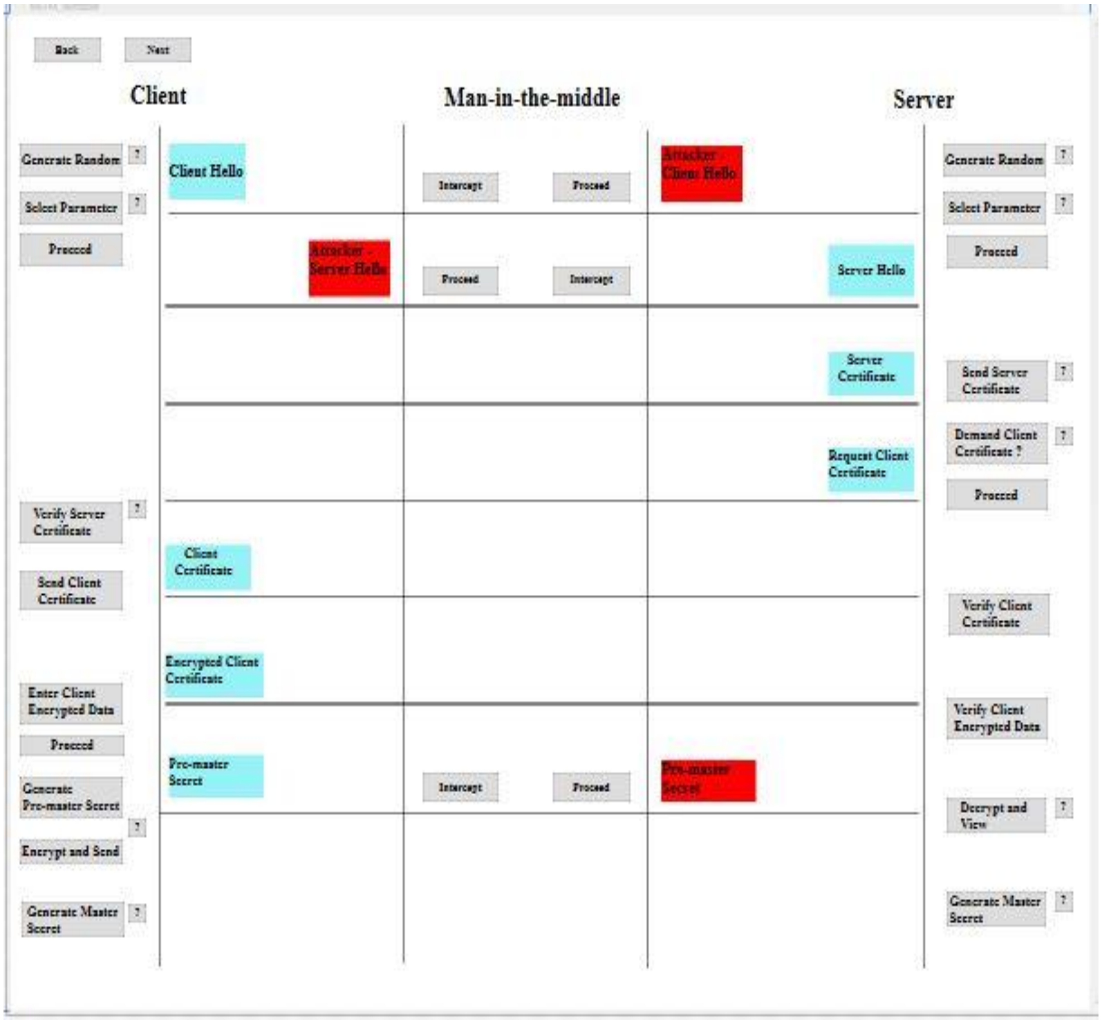


Figure 22: Handshake window for Man-in-the-middle attack

In the similar manner, there are other attacks implemented, like cipher suite rollback attack and version rollback attack. In the cipher suite rollback attack, an intruder tries to intercept the message and modify the cipher suite to weaker ones in order to open the way for several other exploits that might be possible with certain weaker cipher suites.

The same goes with the version rollback attack, where the attacker intercepts the "client hello" message and downgrades the SSL/TLS version, thus allowing several other attacks to proceed.

5.2 Use of library functions in the implementation

This implementation uses several cryptographic library functions provided by MSDN, as well as external security library, called Mentalis.

Table 2 describes the entire usage of libraries in our program by specifying what functionality of the program uses which library function.

Table 2: Library functions used in our program,

Button	**Library used**	**Underlying Class**	**Function defined under the class**

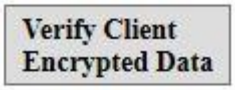
Primary function associated with the button: **Certify()**
	MSDN	RSACryptoServiceProvider
		RSAParameters
			ImportParameters(RSAParameters parameters)
			ExportParameters(bool IncludePrivateParameters)

Primary Function associated with the button: **verifySignature(byte[] signedData, string name)**
	MSDN	RSACryptoServiceProvider
		RSAParameters
			ImportParameters(RSAParameters parameters)
			ExportParameters(bool IncludePrivateParameters)

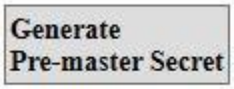
Button	Library used	Underlying Class	Function defined under the class

Primary Function associated with the button: **signData(byte[] DataToSign, RSAParameters key)**
	MSDN	RSACryptoServiceProvider
		RSAParameters
			ImportParameters(RSAParameters parameters)
			ExportParameters(bool IncludePrivateParameters)
			SignData(byte[] buffer, object halg)

		SHA1CryptoServiceProvider	SHA1CryptoServiceProvider()


Primary Function associated with the button: **verifySignature(byte[] DataToVerify, byte[] SignedData, RSAParameters Key)**
	MSDN	RSACryptoServiceProvider
		RSAParameters
			ImportParameters(RSAParameters parameters)
			ExportParameters(bool IncludePrivateParameters)
			VerifyData(byte[] buffer, object halg, byte[] signature)

		SHA1CryptoServiceProvider	SHA1CryptoServiceProvider()


 	
Primary Function associated with the button: **encryptPreMaster()**
	MSDN	RSACryptoServiceProvider
		RSAParameters
			ImportParameters(RSAParameters parameters)
			ExportParameters(bool IncludePrivateParameters)
			Encrypt(byte[] rgb, bool fOAEP)

 Primary function associated with the button: **genmastersecret_client_click()**	Mentalis.org	Ssl3CipherSuites	GenerateMasterSecret (byte[] premaster, byte[] clientRandom, byte[] serverRandom)
		Ssl3DeriveBytes	Ssl3DeriveBytes(byte[]

Button	Library used	Underlying Class	Function defined under the class
			secret, byte[] clientRandom, byte[] serverRandom, bool clientServer) GetBytes(int cb)

Primary function associated with the button: **ApplyHashMACClient()**
	MSDN	HMACMD5
		HMACSHA1
		HMACSHA256
		HMACSHA384
		HMACSHA512
		HashAlgorithm
			HMACMD5(byte[] key)
			HMACSHA1(byte[] key)
			HMACSHA256(byte[] key)
			HMACSHA384(byte[] key)
			HMACSHA512(byte[] key)
			ComputeHash(byte[] buffer)

DataEncryptClient()		DESCryptoServiceProvider	CreateEncryptor(byte[] rgbKey, byte[] rgbIV)
		AESCryptoServiceProvider	CreateEncryptor(byte[] key, byte[] iv)
DataDecryptServer()		DESCryptoServiceProvider	CreateDecryptor(byte[] rgbKey, byte[] rgbIV)
		AESCryptoServiceProvider	CreateDecryptor(byte[] key, byte[] iv)

CompareHashServer()		HMACMD5
		HMACSHA1
		HMACSHA256
		HMACSHA384
		HMACSHA512
		HashAlgorithm
			HMACMD5(byte[] key)
			HMACSHA1(byte[] key)
			HMACSHA256(byte[] key)
			HMACSHA384(byte[] key)
			HMACSHA512(byte[] key)
			ComputeHash(byte[] buffer)

CHAPTER 6: CONCLUSION AND FUTURE WORK

With the development of this application, we can be assured of one thing that, visualizing the whole protocol gives us better understanding in addition to the theoretical knowledge of SSL/TLS. One could use our program to develop a laboratory exercise to teach students about the protocol. Similarly, students after learning about SSL/TLS from the theoretical perspective could use our program as an educational tool to get the clear picture of the protocol. The use of the available security libraries made the flow and implementation of the program easier and dynamic, where writing the entire piece of code from scratch was replaced by the library function call that provided already built in functionality.

As it is the case for every newly developed application, there are certain limitations of the current version of the program. The implementation of the protocol is purely for educational purposes and thus cannot be used in the actual browser for the communication. Also, our implementation cannot be used to exchange data between two machines using socket communication. Being developed with Microsoft IDE (Visual Studio), this application might not be platform independent and may only work on Windows.

Considering some features and functionalities, the attacks demonstrated in this program, such as cipher suite rollback and version rollback, could be used as a base for some other exploits that are more complex to implement. This attacks includes POODLE attack, which is based on the version rollback attack, but is really very complex and difficult to demonstrate as a part of our program.

Furthermore, I have been able to implement the handshake and the record layer protocols, however as the future work is concerned, this development could also be extended by implementing the "alert protocol" and the "change cipher spec protocol," which would complete the entire SSL/TLS implementation.

REFERENCES

[1] W. Chou. “Inside SSL: the secure sockets layer protocol”. IT Professional, vol. 4, no. 4, pp. 47–52, Jul. 2002

[2] T. Elgamal. “The Secure Sockets Layer Protocol (SSL)”. Internet: http://www.ietf.org/proceedings/95apr/sec/cat.elgamal.slides.html, Apr. 1995 [Mar. 03, 2003]

[3] J. K. Harris. “Understanding SSL/TLS”. Internet: https://computing.ece.vt.edu/~jkh/Understanding_SSL_TLS.pdf, Oct. 2008 [Sep. 16, 2014]

[4] Introduction to SSL,” *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Introduction_to_SSL.

[5] What is the difference between SSL 2.0 and 3.0?” [Online]. Available: http://stason.org/TULARC/security/ssl-talk/4-11-What-is-the-difference-between-SSL-2-0-and-3-0.html#.VVtZw0beM6l.

[6] Differences between SSL and TLS Protocol Versions.” [Online]. Available: https://www.yassl.com/yaSSL/Blog/Entries/2010/10/7_Differences_between_SSL_and_TLS_Protocol_Versions.html.

[7] H. Krawczyk. “The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)” in Advances in Cryptology — CRYPTO 2001. J. Kilian, Ed. Springer Berlin Heidelberg, 2001, pp. 310–331 [11] Martin, Franck. “SSL Certificates HOWTO”. Internet: http://www.tldp.org/HOWTO/SSL-CertificatesHOWTO/, Oct. 20, 2002 [Mar. 14, 2003]

[8] SSL: Foundation for Web Security - The Internet Protocol Journal - Volume 1, No. 1,” Cisco. [Online]. Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html.

[9] S. Adamovic. Video Lecture. Topic: “Authentication Using Public Keys – Lab 3”. Nov. 06, 2012

[10] System.Security.Cryptography Namespace (). [Online]. Available: https://msdn.microsoft.com/en-us/library/system.security.cryptography(v=vs.110).aspx.

[11] SSL, TLS and PCT classes for C# and VB.NET. [Online]. Available: http://www.mentalis.org/soft/projects/ssocket/.

[12] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0." [Online]. Available: https://tools.ietf.org/html/rfc6101

[13] SSL handshake, online, available at, http://commons.wikimedia.org/wiki/File:Ssl_handshake_with_two_way_authentication_with_certificates.png

[14] SSL/TLS in Detail, http://technet.microsoft.com/en-us/library/cc785811%28v=ws.10%29.aspx

[15] Illustration Image Reference, https://vpnreviewer.com/wp-content/uploads/2015/01/insecure-email.jpg

[16] Introduction to SSL Image Reference, http://www.goclio.com/2009/06/19/10-things-every-lawyer-should-know-about-legal-saas-part-4-security/

[17] C# tutorial, http://csharp.net-tutorials.com/

[18] C# WPF video, https://www.youtube.com/watch?v=krxYDsee2cQ

[19] SSL Handshake steps in detail, www.pierobon.org/ssl/ch2/detail.htm

[20] POODLE - Definition, detection and remediation." [Online]. Available: http://learning.criticalwatch.com/poodle/

[21] CRYPTOOL PORTAL - CRYPTOOL1 - HOME." [Online]. Available: https://www.cryptool.org/en/cryptool1-en

[22] CRYPTOOL PORTAL - CRYPTOOL2 - HOME." [Online]. Available: https://www.cryptool.org/en/cryptool2-en

[23] CRYPTOOL PORTAL - CRYPTOOL-ONLINE - HOME." [Online]. Available: https://www.cryptool.org/en/cryptool-online-en

[24] CRYPTOOL PORTAL - JCRYPTOOL - HOME." [Online]. Available: https://www.cryptool.org/en/jcryptool-en

[25] GnuPG Tutorial." [Online]. Available: http://bitflop.com/tutorials/gnupg-tutorial.html

[26] GNU Privacy Guard," *Wikipedia, the free encyclopedia*

[27] MAGMA.” [Online]. Available: http://icl.cs.utk.edu/magma/

[28] Magma (algebra),” *Wikipedia, the free encyclopedia*

[29] Magma Computational Algebra System.” [Online]. Available: http://magma.maths.usyd.edu.au/magma/

[30] SageMath - Download Binaries for Microsoft Windows.” [Online]. Available: http://www.sagemath.org/download-windows.html

[31] SageMath - Source Code Distribution.” [Online]. Available: http://www.sagemath.org/download-source.html

[32] SageMath,” *Wikipedia, the free encyclopedia*

[33] SageMath, Inc.” [Online]. Available: http://www.sagemath.com/

[34] SageMath Mathematical Software System - Sage,” *SageMath Mathematical Software System*

[35] ECE646_lecture11_hash_MAC_2.pdf.” [Online]. Available: http://ece.gmu.edu/coursewebpages/ECE/ECE646/F14/viewgraphs_F14/ECE646_lecture11_hash_MAC_2.pdf.

[36] SSL/TLS Strong Encryption: An Introduction - Apache HTTP Server Version 2.4.” [Online]. Available: http://httpd.apache.org/docs/2.4/ssl/ssl_intro.html

[37] Why use Ephemeral Diffie-Hellman - Knowledge Base - mbed TLS (Previously PolarSSL).” [Online]. Available: https://tls.mbed.org/kb/cryptography/ephemeral-diffie-hellman

[38] Web application - What key exchange mechanism should be used in TLS? - Information Security Stack Exchange.” [Online]. Available: http://security.stackexchange.com/questions/8343/what-key-exchange-mechanism-should-be-used-in-tls

[39] An Introduction to OpenSSL, Part Four: The SSL and TLS Protocols | Symantec Connect.” [Online]. Available: http://www.symantec.com/connect/articles/introduction-openssl-part-four-ssl-and-tls-protocols

[40] Working with App.xaml - The complete WPF tutorial.” [Online]. Available: http://www.wpf-tutorial.com/wpf-application/working-with-app-xaml/

[41] WPF Tutorial : Beginning - CodeProject.” [Online]. Available: http://www.codeproject.com/Articles/140611/WPF-Tutorial-Beginning

[42] Differences between SSL and TLS Protocol Versions.” [Online]. Available: https://www.wolfssl.com/wolfSSL/Blog/Entries/2010/10/7_Differences_between_SSL_and_TLS_Protocol_Versions.html.

[43] Security - What are the exact protocol level differences between SSL and TLS? – Server Fault.” [Online]. Available: http://serverfault.com/questions/178561/what-are-the-exact-protocol-level-differences-between-ssl-and-tls

[44] SSL versus TLS - What’s the difference?, *LuxSci FYI*.

[45] About SSL/TLS.” [Online]. Available: https://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS8a/SSLTLS.html

[46] A brief chronology of SSL/TLS attacks | Java Security and Related Topics.

[47] Why is TLS susceptible to protocol downgrade attacks? - Cryptography Stack Exchange.” [Online]. Available: http://crypto.stackexchange.com/questions/10493/why-is-tls-susceptible-to-protocol-downgrade-attacks

[48] OpenSSL: The Open Source toolkit for SSL/TLS.” [Online]. Available: https://www.openssl.org/

[49] OpenSSL,” *Wikipedia, the free encyclopedia*

[50] OpenSSL: Source, Tarballs.” [Online]. Available: https://www.openssl.org/source/

[51] OpenSSL PKI Tutorial v1.1 — OpenSSL PKI Tutorial.” [Online]. Available: https://pki-tutorial.readthedocs.org/en/latest/

[52] OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs,” *DigitalOcean*. [Online]. Available: https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs

[53] Cryptography - What are the differences between the versions of TLS? – Information Security Stack Exchange.” [Online]. Available: http://security.stackexchange.com/questions/705/what-are-the-differences-between-the-versions-of-tls

BIOGRAPHY

Harsh Vachharajani was born in October of 1991 in Gandhinagar, Gujarat, India. He graduated from Infocity Junior Science College, Gujarat, India, in 2009. He received his Bachelor of Technology in Information Technology from Ganpat University, Mehsana, India, in 2013. He further continued his studies pursuing his Master of Science degree in Information Security and Assurance from George Mason University, Fairfax, USA, in 2015. During his course of studies, he worked as a Research Assistant in the Cryptographic Engineering Research Group (CERG). His research interest include software implementation of cryptographic algorithms and protocols, and vulnerability assessment for securing software applications.