

IMAGE PROTECTED - HYPERTEXT TRANSFER PROTOCOL

Aryan Chandrakar ¹

Under Guidance of Dr. Sureshkumar N ²

¹ School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, India

² Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, India

Abstract – We protect the data using encryptions. Data although being encrypted can be seen by anyone and with proper logic, resources and time one can easily break the encryption within given period of time. A solution to this problem lies in the steganography, an art of inconspicuously hiding the data well enough that unintended recipient doesn't suspect even a pinch of the data hidden right in front of the eye. Steganography isn't a new concept, it came to existence during the golden ages of roman empire, where the inscription on the wooden tablet were hidden using wax coating to look like normal wood [1]. The internet is a massive playground for the modern techniques to be discovered and implemented, example think of IP packets as was tablet and the packet data is the inscriptions on the wooden tablet. The headers serve as the wood in the analogy. The well-known SSL layer being used in the https communication on daily basis is said to be secure, encrypting the content related to user's and server's response and request. It works on the basis of DHE with AES-256, where DHE is Diffie Hellman with is the base of the whole structure on which the security lies, encryption has become easier to be solved using various kinds of online tools. While the security side of the same might look promising but man in the middle attack can facilitate in making is possible to bypass the same. The model of steganography provides us with a solution to hide the data in an image, sound or other piece of data, the attacker won't be able to attack the data whether encrypted or not, if it can't be seen. This undermines the possibilities of many different attacks making it much safer to communicate and connect.

Keywords - Communication, Information Security, Image Processing, Man in Middle Attack, Steganography, Web-Security

I. INTRODUCTION

Data to a normal human eye usually contains images, sounds and text. Internet users frequently need to send, store and receive private information. The most commonly used way to achieve the goal is to change the data into some other format, thus the resulting data can only be understood by those who know how to return

the information back to it's original form. This method of protecting information is known as encryption. The major drawback of encryption is that the data is not hidden. Information although hidden using various levels of encryption is clearly visible to everyone, and anyone with proper determination and resources is completely and fully capable to break the information provided the time bounds doesn't apply in the situation.

An answer to the problem lies in the game of Hide and Seek to remove the 'thing' from the plain sight beforehand such that there is nothing to start with at the first place. Image steganography or hiding the data within the image using different methods like hiding between the bit slices of the image. Combining the technique with the normal IP packet serving as the tablet for the basic necessary information that would be redundant in relation to the information in the packet or information hidden in the image.

The purpose of steganography is not just to keep people from knowing the hidden information but to make them think there is any existing information in the first place. Steganography relies on the naivety of human beings. Both encryption and steganography had to be used to get the best result as the security is about both the aspects privacy and protection, where the encryption do provide privacy of the data, steganography helps in providing the needed protection.

The paper's idea works on the same idea of encryption and steganography but prior to all these, it would be discussing about different ways in which the current communication and connection protocol used to establish the connection between server and client falls back and how can it be broken in several ways. Further discussing the particular easier way of MIM attack to break the usually used HTTPS protocol which relies on Diffie-Hellman RSA with AES encryption, of which the encryption part using AES has already been proven to be redundant for stopping a hacker.

Further the work builds on the idea of introducing steganography in the HTTPS protocol as an add-on to provide the needed protection against the encryption

breakage and MIM attacks. Thus introducing a protocol basically given the name HTTPPI, which relies on the content being shared between server and user be encrypted and hidden using image processing – steganography, while sharing the image the attacker won't be able to attack as man in the middle, cause the image is encoded with data using a shared key between the server and client, which being a large prime number is hard to guess and the value changes on each connection. This will undermine the possibility of not only man in the middle attack but also provide a safer way to communicate.

While keeping in mind the necessary need of privacy of data and to hide the basic pattern of steganography data in the LSB layer of the bit plane slice the framework introduces RSA algorithm to provide the needed privacy, as RSA is recommended to be used with long prime numbers in the range larger than billions too, or to say of the order of 2048 bits as a minimum criteria length of the numbers. This will considerably provide the data with an encryption the chances of breaking of which is less than the chances of earth getting hit by meteor in the next two seconds and get destroyed i.e. unbreakable encryption unless hacker is lucky enough to guess a minimum 2048bit length prime number with the exact precision provided within the time limit.

II. LITERATURE SURVEY

Steganography isn't a new concept, it came to existence during the golden ages of roman empire, where the inscription on the wooden tablet were hidden using wax coating to look like normal wood. The internet is like free ground for a kid, witing to be explored, discover new techniques and implementation, considering the case of IP packets as tablet the tablet of our story and the packet data is the inscriptions on the wooden tablet. The purpose of steganography is not just to keep people from knowing the hidden information but to make them think there is any existing information in the first place. Steganography relies on the naivety of human beings.

Encryption and steganography achieve distinct goals for the better, encryption encodes data changing the look of the same, such that an unintended recipient can't make anything out of it, while steganography tries to prevent an unintended recipient from getting any idea about the data presence, hiding in the plain sight like a camouflage. As privacy concerns continues to evolve along with the digital domain, steganography might turn out to play an important role in the coming future, software would be easily able to transmit user's sensitive information without any second person's permission or knowledge.

Besides the lack of attention given to steganography, the field presents a few problems whose solution will have a considerable effect on the internet and communication. [1]

Several ways are present to hide secret bits within the message, of these various ways the most secure, high performance image system were based on three approaches, basically choosing suitable cover image, select appropriate embedding location, and the kind of data to be embedded, which might result in secure system that might defeat several attacks and analysis. [2]

Two systems came to design for image steganography with JPEG compressed images, the prior based on ST-SCS watermarking and the latter based on histogram preserving data mapping HPDM. The ST-SCS model was not always accurate, and many assumptions were made to keep the complexity of the model at acceptable level. The input image needed to be compressed with a factor of 75. The data embedding rate can be modified by the user based on spreading factor of the spread transform. A spreading factor of 1 gave the highest data rate but produced many distortions while no side information had to be provided or transmitted to the decoder. While on the other hand the HPDM based scheme provided error free communication without any distortion, but on the other hand a prior side information has to be transmitted in the first communication subchannel to the decoder, for which the encryption was utilized in order to resist the steganalysis.

Both the design model had pros and cons, however ST-SCS leaves a noteworthy trace in the statics as the cover image comes out to be not smooth, with a bump in result data at the embedding locations. The HPDM design can be considered secure, with an easy and straight forward, providing security even against the classiest steganalysis. [3]

Sometimes these detections come out to be false positive, wasting time and raising alarms, to solve the issue a passive warden comes to play who doesn't alter images but keeps a check on the bit which becomes a problem in case of data hiding using steganography, LSB based steganography is the most simple and straight forward approach against the passive warden. To add if the message is embedded directly into the least significant bit slice plane of the cover image, then the resultant structure of the bits in the LSB would clearly be an indication to the third party giving away the main motivation behind whole approach. Thus, to maintain the random appearance of the LSB, the message comes to be encryption for a while before any further steps. The best

approach to be safe from the structure detection is to hide the data in encrypted form dissolving the whole structure of the data. [4]

Practices to hide meta-information about a message such as duration, sender and receivers are together known as traffic security. Steganography is often used as a small part to achieve the goal of that security. Theoretically the perfect secrecy doesn't exist, none consider the entropy of data which leave us with quantitative leverage combining with the bandwidth of the stego-key gave away the suggestion of embedding data in parity bits rather than data directly, which would even allow us to carry out public key protected steganography. [5]

Even though distributed dictionary attack was implemented using the Stegabreak while it was too slow to run the attack which won't be enough to run the attack in enough time to get to know the content being shared before the sender or the receiver gets to know about the same. To add in, the fastest rate was 12504 words per second, the attack was able to guess the password only those which were not strong or to say 25% of all the password aka the vulnerable passwords only, which leaves us looking at either wrong place or widespread use of steganography. [6]

Any particular website communicating between client and server or between two nodes could be blocked from upgrading HTTP to HTTPS. While more than half of the sites loaded on HTTP rely on the active or passive HTTP resource that is not available in HTTPS. Websites dependency on other sites nearly double in the last five year and only negligible is free of this reliance. One third of the top million sites provides user with unknown content from the ad networks and the third parties, which brings in the slap of drastically increase in the attack surface with a kiss of decreasing the load on the main site. Not only does this tangle with the dependencies, but also introduces the host site to security and privacy concern that were never there before, and this increased complexity sometimes blocks the website from migrating to HTTPS. [7]

When a communication is started between the client and the server by the specifying URL using the port number 443, assigned to HTTPS, the web server sends a certificate to the client. Through this procedure the browser specify the public key the conversation needs to be with, or to say in short the procedure to form the secure connection between the client and server is based on Diffie-Hellman key exchange protocol. Technically, the initial data is the key which is used during further data transmission. Both the client and the webserver use

separate private keys to decode the information transmitted among them. While the whole procedure falls under question if a man in the middle attack is performed, which might result into delay in communication between client and server, but this delay could be given the name of various possibilities which help the attack to masquerade itself.

By this, all packet transmitting between the victim and the server passes through the hacker machine. Once the victim requests to server in order to send the information on secure connection mentioning HTTPS, server generates certificate. The hacker, consequently, captures the certificate and sends the other side new fake certificate including hacker's public key. Every time the victim requests a response from server, the information is actually stopped at the hacker machine, according to Man in the Middle analogy. [8]

Diffie-Hellman and RSA operations are the main components in any communication between client and server. In both the client i.e. the victim has to compute large values up to several powers of a number and the computation time for the same can be exploited by the attacker to bypass the protocol, the timing becomes a giveaway when considering the key size the amount of computation time can be a sign towards the key size and exponent bit counts and a modular algorithm can be designed to find the keys using the same amount of time decreasing the large key dictionary to less than tenth of the same.[9]

It's also possible to attack the HTTPS secured connection by exploring the properties of common LAN as well as typical behaviours of inexperienced users but might be difficult in case of an experienced user. The ease with which an attacker can spoof the whole system underlines the fact and responsibility that sites should consider the potential hazards of self-signed certificates. Although strong encryption is powerful in providing data protection, the security is only as good as the weakest link in the chain. [10]

Problem Statement: Inspired from the ancient roman tablet used for communication, without anyone having any suspicion about the information being there at the first place. The approach used in this paper is based on to break the limits of communication protocols that are used on daily basis. Breaking the communication protocol using spoof and MITM attack without raising any alarm the victim's communication channel is compromised. While not stopping at the compromising the channel a better version will be proposed in order to provide user with safer way to communicate. All these

approaches are going to be tested in this paper and be compared with each other in order to reach the best possible solution.

III. PROPOSED RESEARCH FRAMEWORK

Forming a new communication protocol has various attributes that tags along with the same, but the prior task would be to show that current communication protocol has faults and lies behind in certain aspects, followed by proposal of a version which deals with the left-out aspects in the earlier method. The implementation is in two folds the first fold is to bypass the HTTP & HTTPS with a trial for HSTS bypass too by spoofing the target in order to change the already existing site table in the browser when considering the HSTS. And the second is to introduce a new method in order to provide a safer communication channel which would be looked into with consideration of image processing for the rescue and providing a safer way.

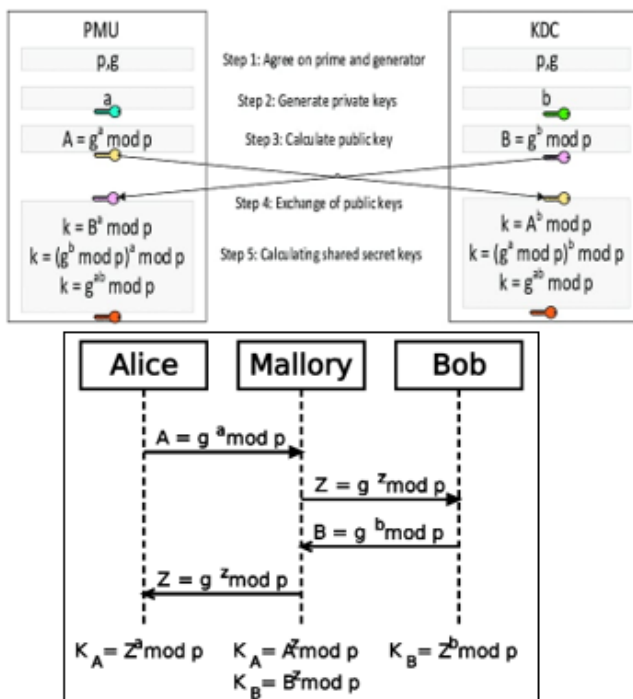


Figure 1 a) Diffie-Hellman process b) breaking DH with MITM attack

The well-known SSL layer being used in the https communication on daily basis is said to be secure, encrypting the content related to user's and server's response and request. It works on the basis of DHE with AES-256, where DHE is Diffie Hellman with is the base of the whole structure on which the security lies, encryption has become easier to be solved using various kinds of online tools. While the security side of the same

might look promising but man in the middle attack can facilitate in making is possible to bypass the same.

The idea for the paper came from the Cicada case using image processing for communication.

The proposed method lies on the basis of challenging the issue and bringing forward the model of HTTPPI, which relies on the content being shared between server and user be encrypted and hidden using image processing – steganography, while sharing the image the attacker won't be able to attack as man in the middle, cause the image is encoded with data using a shared key between the server and client, which being a large prime number is hard to guess and the value changes on each connection .

The LSB based steganography either changes the pixel value by ± 1 or leave them unchanged, which depends on the nature of the hidden bit and the LSB of the corresponding pixel value where the final pixel value can be found as $I = \{x_i, i \in Q\}$ where Q is an index set denoting the mean of subtracted cover image.

This will undermine the possibility of not only man in the middle attack but also provide a safer way to communicate.

IV. ARCHITECTURE

I. Overview

As discussed in the earlier section the implementation is divided into two sections based in different concepts together focusing on evolution of communication protocols. To achieve the goal each section is tested separately, with the first section being carried out on Linux interface the second section is completely coded on python interface for easier implementation and availability of libraries to carry out the construction. The sections are divided on the basis of breaking and strengthening of the protocols.

II. Breaking the communication protocol - Man In the Middle Attack

The section deals with breaking and bypassing the current communication protocols. This attack can be launched when we are able to intercept the communication between two devices, hence the name man in the idle attack MITM. Where a normal communication would look like shown in fig.2a where the device is directly communicating with the entity they want to communicate with. But in MITM the adversary would be able to place themselves in the middle of the connection allowing them to intercept and see anything

that is being transferred between the devices as shown in fig.2b.

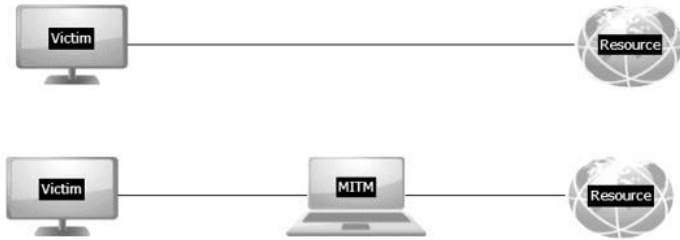


Figure 2 connection between server and client a) no MITM b) with adversary as MITM

There are number of methods to achieve this, the method we will look into is ARP spoofing attack, which allows us to redirect the flow of packets so instead of it flowing as shown in the fig.3a it would flow through the adversary's computer, thus all the request sent and response received by the target computer will flow through the adversary's computer i.e. any message, websites, image, username and password entered by the target will have to flow through adversary's computer allowing them to read the information, modify it or drop it. This is possible because the ARP used is not very secure.

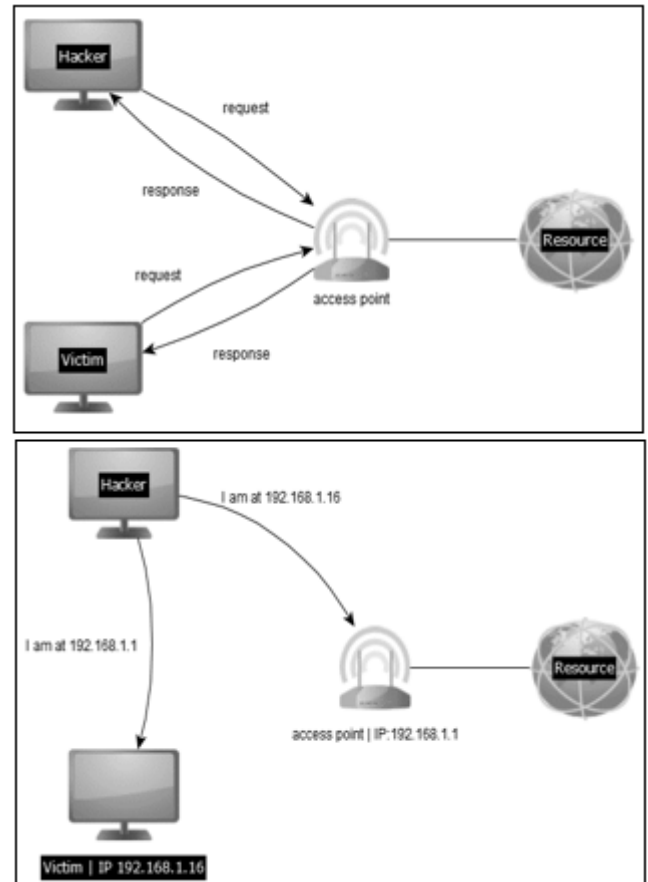


Figure 3 a) normal network connectivity b) connectivity during MITM

ARP stands for address resolution protocol and is very simple protocol that allows us to link IP addresses and MAC addresses using broadcast messages, by responding the SRP request for an IP address with a conformation and the MAC provided in the response to associate them together and store the table in each computer on network. But the MAC address can be easily affected by exploiting the ARP protocol as shown in fig.3b. Thus any time a request and response is sent is going through adversary's machine.

The reason for all this being possible is that client can accept responses even if they didn't send any request, so even if we send a response to access point gateway and the victim with the message specifying false addresses, they will accept the response without verification. There are several ways to poison the ARP table but as shown in the next section we would be focusing on ARP spoofing using both arpspoof command on Linux interface and Bettercap with use of caplets to bypass the communication protocols.

III. Strengthening the protocol – Image Processing Steganography and RSA encryption

Divided into three sections the strengthening protocol works as to automatically capture the packet data containing all the important message o request and response, further encrypting it using RSA which till date has no published method to be defeated. And finally protecting the encrypted data using LSB based image steganography.

1. Capturing packets

Based on scapy request and response reading and accessing the approach is simple we sniff a particular port using scapy module, without storing any data or request, further processing the packet based on layers in requests, if the layer has any HTTP/HTTPS requests it is captured and sent to the next section for processing.

2. Encrypting packet's data

RSA is an asymmetric cryptographic algorithm using two different keys for encryption i.e., private, and public key. The algorithm is based on the idea that it's difficult to factorize a very large number or in our case 2048bit long key, made up of two numbers which are multiplication of two large prime numbers. As the private key is derived from the same two numbers considered earlier, the strength of the private key lies in the strength of the prime numbers considered. Experts believe that 1024-bit size keys might be compromised in the near future thus we would be considering only 2048 bit keys for implementation.

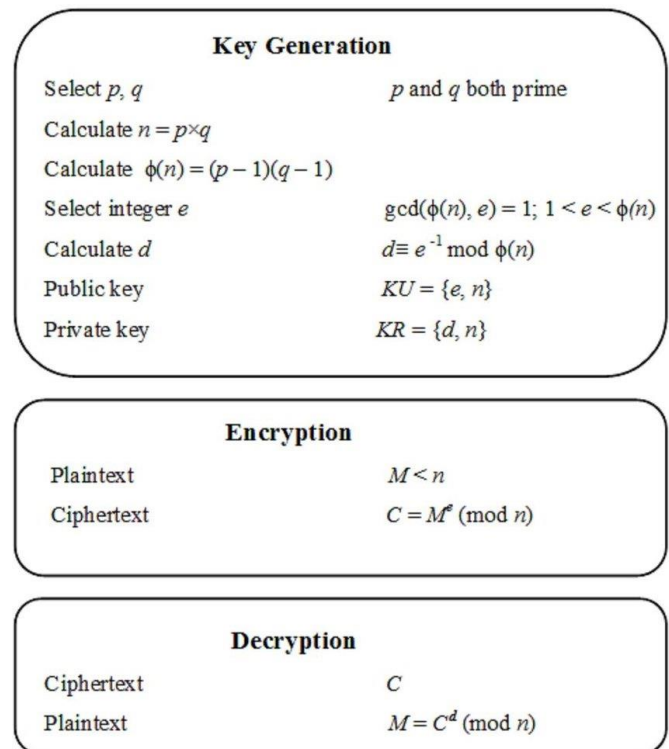


Figure 4 RSA algorithm [15]

3. LSB Based Steganography

Hiding data in images using the least significant bit technique is divided into two subsections encoding and decoding, which are further divided into different steps and phases.

3.1. Encoding

The encoding phase starts with check of the image size to be compatible enough to hold all the data, once checked the data of the image is converted into binary values including the width and height, upon which the iteration of slots will occur to store the data. Next, the pixel values of the image are stored in a list and based on the bits of the data the pixel's value, converted to binary value and the LSB layer of the image are either in 1s and 0s as shown in fig.5, are changed to 0 or 1 depending on the initial values and the cursor moves to the next slot. Once the bit change is performed on the layer the binary values are converted back to the initial bit size.

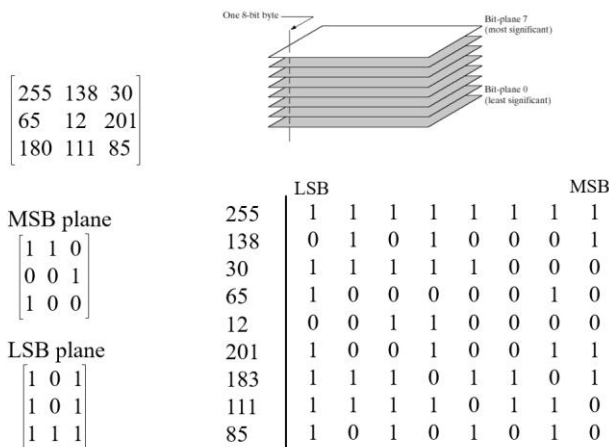


Figure 5 Bit plane slices with example of 3x3 matrix conversion to bit planes

While even after changing the binary values in the LSB layer it won't have any effect on the final image as the LSB layer with or without plays almost negligible role in the image formation as shown in fig.6. The image when broken into layers and combined back together the combination of just bit plane 7,6,5 and 4 gives almost similar image as original as shown in fig.7 thus the effect of making changes in the LSB layer which looks like salt and pepper noise won't have any affect unless one performs encoding in a pattern form which would surely be a give-away.

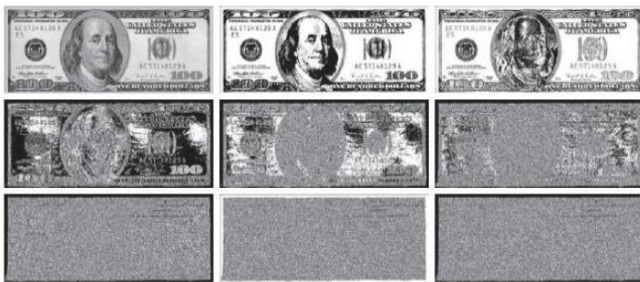


Figure 6 Original image and 8-layers of bit plane slices[14]



Figure 7 bit plane slice combinations of planes {7, 6}, {7, 6, 5}, {7, 6, 5, 4} [14]

3.2. Decoding

The decoding phase is much simpler than the encoding phase, which starts with first counting the number of bits to be read in the image, once the size is noted each bit is started reading and based on the binary values of these bits the function returns either 0 or 1. Once a value is

returned the reading slot is moved to next bit for further processing. Once all the values are returned by the function the binary value noted in the whole process is converted to byte for user understanding.

4. Decrypting fetched data

As discussed in the encryption section the RSA being asymmetric key encryption, would be using a public key for encryption at the sender's end while a private key, associated with the public key, is used to decrypt the data. Anyone with the public key can encrypt the data but no one can decrypt the same without the private key which is secret to the receiver only.

The public key is shared between the two ends as usual as seen in the HTTPS protocol thus no further change would be required for this layer. Just the change for key size might be required when connecting between the clients as the safer side to be on is to use a more than 2048bit sized key.

V. IMPLEMENTATION

The implementation of the proposed architecture would be discussed in the phases as discussed in the earlier section. The whole implementation was carried out on Linux platform with use of Linux commands and python programming to achieve the required test phase which is discussed in next section.

I. Breaking communication protocol- MITM attack

To spoof the target to view oneself as another gateway through which the packets pass, arpspoof is used to fool the target in thinking so. Arpspoof is an inbuilt command on Linux interface which requires the interface on which the target and adversary's system is connected and the target's client IP address and gateway IP address which could be easily fetched using the arp-a command which can help one discover all the link layer addresses like MAC address, IPv4 address and the gateway the system is connected to.

```
arpspoof -i [interface] -t [clientIP] [gatewayIP]
arpspoof -i [interface] -t [gatewayIP] [clientIP]
```

code 1 arp-spoof commands

But this would result in only providing one a way to spoof themselves as the gateway between the client and server, in order to capture packets and play with the same, the use of Bettercap has been emphasised, which is a powerful tool created by the sole purpose of various kinds of MITM attacks against network, manipulate HTTP, HTTPS and TCP traffic in real-time, sniff all the

sensitive information like password, username and much more and also providing a way to downgrade any website from HTTPS to HTTP by changing the port address from HTTPS's 443 to HTTP's 80. Bettercap requires the interface as parameter on which it would be running and one can perform all kinds of actions after starting the same including spoofing the target, probing, use caplets to downgrade website and sniff data on the network.

```
Bettercap -iface [interface]
> net.show - to get all the Ips
> net.probe on - to start
monitoring the packets
> set arp.spoof.full duplex true
- if set to true, both the target
and gateway will be spoofed, as
for gateway spoofing as target and
for target as gateway, else only
the target
> arp.spoof.targets [target name]
- comma separated list of all Ips
and MAC addresses to spoof
> arp.spoof on - to turn on
spoofing on & off
```

code 2 Bettercap usage

All the above can also be converted to one command by the use of caplets, a text file with all these commands saved with .cap extension accessed while starting Bettercap.

```
bettercap -iface wlan0 -caplet spoof.cap
```

code 3 Bettercap usage with caplets

Once the command start running adversary can downgrade the website by using sslstrip which comes as another predefined caplet, a part of Bettercap and can be seen using the caplet.show command.

II. Encrypting - Decrypting data to be hidden

In order to protect from the attack implemented above the plan to use image processing came up at the first place but before hiding the data in the image's LSB layer the data has to be encrypted else the pattern of the bit values would show a pattern to a well-trained eye which would be the giveaway of whole introduction of image processing in the first place. Thus, to hide the data perfectly the data is encrypted using the RSA asymmetric key encryption, which come as predefined library in python under the library name Crypto. Crypto offers the

RSA encryption process using just a single line and the data can be padded to the required condition of RSA using PKCS1_OAEP offered under the same Crypto library.

Once imported the RSA module is used to generate the key pair of public and private key of needed length. Before encrypting the data using the public key the data is UTF-8 encoded and given to encrypt class of the RSA module as input, further after encrypting the same using the public key it is sent to be hidden in the image.

As the image is shared the fetched data needs to be decrypted, in order to do the same the private key associated with the public key is used with the encrypted data as the input the decrypt class of RSA module gives the human readable data back.

III. LSB based steganography

Encoding phase, where the data and image is converted to their binary value as shown in fig.5, and as discussed in the previous section to achieve the same the whole process occurs in various phases.

Encode Text

Before any operation can be performed the whole text data is encoded into its byte value corresponding to the character of the string. The byte value, 8-bit long binary values, are used further to encode the data in the image.

Binary values - image

Converting to the binary value of the image integer pixel values, in case the length of binary value is greater than the bit-size expected an exception is thrown and in case if the binary values are less in number, they are padded with '0', finally returning the required value for further usage.

Adding the data binary values onto image

The pixel values are provided to the function in a list and based on the bit value of the image the pixel values are altered at the LSB layer. If the value of the data is already '1' or operation is performed with the pixel bits else and operation is performed. The resulting values are stored in a tuple such that no further changes can be made for security purposes.

Next slot

Once the operation is performed the window on which the operation is being performed by the cursor is moved forward and in case the column is completed the cursor moves to the next column's first row to perform the

operation. In case the data is left and all the slots are already written over an exception is thrown about no slot being available else the procedure carries on until whole data is encoded.

Byte value

Once the whole above process is completed the resulting 8-bit binary value is converted to its integer byte value such that it can be used back again in the image.

Decoding phase, where the data is fetched from the image in order to provide the user with the hidden content similar to encoding phase occurs in phases to provide with the needed data.

Read bits

Once the number of bits in the are noted, each single bit in the image is read at a time and the binary value of the same undergoes and operation with '1', the resulting value is sent to further function for processing.

The same thing occurs for all the slot window in the image as the cursor moves forward using the "next slot" operation as discussed earlier.

Decode byte to text

Once all the bits on the image are read they are to their 8-bit byte value sets and send to be decoded in order return the human readable text. The text size is read in bytes and converted to it's corresponding character value which is concatenated together to form the string.

VI. TESTING

The whole construction and implementation of the scenario discussed above was carried out and tested using VMs without affecting any real-world nodes and network. While the adversary runs on Debian (64-bit version) Kali Linux as host OS the victims are attacked on both MS-Edge (Windows10), all the nodes run on 3GB RAM, 64-bit processor, connected to NAT network, 4 processors and PIIX3 chipset. All tests were performed in real-time with base hardware configuration of the system on which everything being conducted- Intel Core i7-8750H CPU @ 2.20GHz, 8GB RAM, 64-bit Operating System, x64-based processor and dual GPU- Intel UHD Graphics630 and NVIDIA GeForce GTX 1050Ti with 128GB SSD.

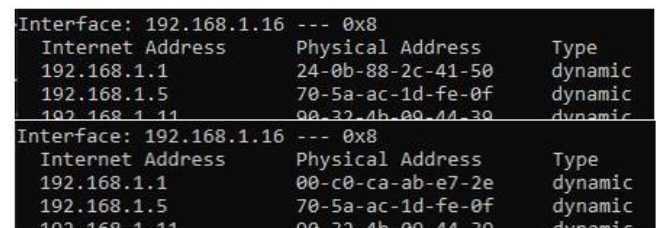
VII. RESULT & DISCUSSION

The result of the tests has been divided into three sections based on the implementation phases itself the MITM results, encryption-decryption results and

steganography results, and comparing them with the current scenario.

I. Breaking communication protocol- MITM attack

Starting with the arp-spoof stage in order to fool the target and gateway into thinking the adversary as gateway and target respectively the arp-spoof was carried out using the arpspoof command. While the results of the arp table were noted before and after the attack as shown in fig.8. The mac address of the gateway has been spoofed to the adversary's mac address such that all the packets pass through their system.



Internet Address	Physical Address	Type
192.168.1.1	24-0b-88-2c-41-50	dynamic
192.168.1.5	70-5a-ac-1d-fe-0f	dynamic
192.168.1.11	00-32-4b-00-44-30	dynamic

Internet Address	Physical Address	Type
192.168.1.1	00-c0-ca-ab-e7-2e	dynamic
192.168.1.5	70-5a-ac-1d-fe-0f	dynamic
192.168.1.11	00-32-4b-00-44-30	dynamic

Figure 8 Target's ARP table before and after spoof attack

While this only provides the opportunity for making the packets pass through the system and read it, in order to perform other actions on those packets like sniffing data and downgrading the website we discussed about using Bettercap. The Bettercap though offers several caplets based on spoofing, injection, sniffer, rogue-MySQL-server etc. but one can create their own caplets and on implementing using the caplet discussed in the earlier section would result in downgrading the website using ssl-strip as shown in the fig.9 for the websites like LinkedIn and Stackoverflow.

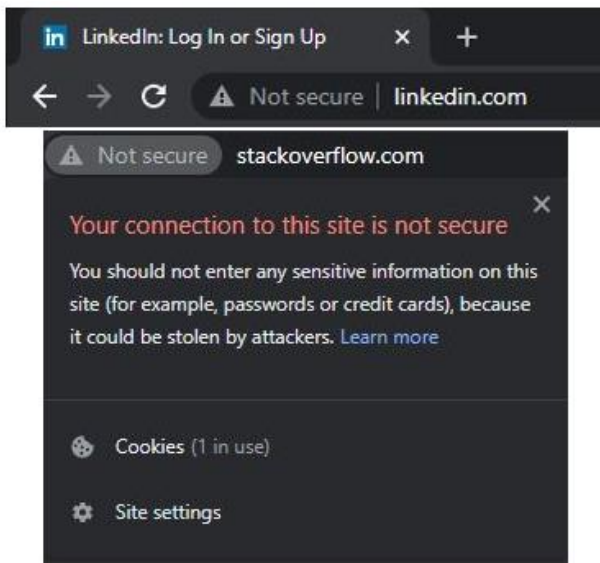


Figure 9 Downgrading websites using Bettercap

Once the websites are downgraded it's a simple job to sniff the packet and fetch the details from the packets including the encoding, content type and credentials like username and password as shown in fig.10.



Figure 10 Sniffing credentials using Bettercap

To be safe from such occurrence the image protected method was proposed, but before diving into the usage of the, we would look at how the data is encrypted before being encoded in the image.

II. Encrypting - Decrypting data to be hidden

For the purpose of encryption and decryption of the data RSA has been employed, as for the security reasons we went two steps ahead to using 3072-bit length key, instead of 1024 which is not recommended as based on the current computing power it would be broken in upcoming future thus 2048-bit length is recommended, while a further step was taken. As discussed, the use of Crypto module does most of the processing providing us with values of n, private key and public key, while the p & q used to compute n, as shown in fig.11a, were hidden.



Figure 11 Sample values of a) N b) Public key c) Private key in RSA

While using the keys and the implementation as shown in fig.11b,c, the message was encrypted using the public key and sent to the other side where the private key is used to decrypt the message.

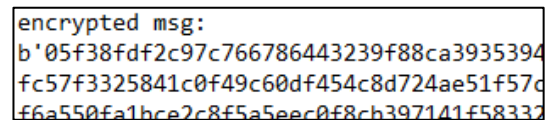


Figure 12 Encrypting data using public key

This encrypted message is used in the encoding process in the steganography process in order to hide the obvious pattern of the message and to look random on the LSB layer just like another bit value.

III. LSB based steganography

The code was executed perfectly with the encrypted data being encoded onto the image, as shown in the figure below the difference between the input image and output image before and after encoding, the difference can't be pointed with an un-aided eye, while even going for the steganalysis like histogram analysis or kurtosis, as the data is encrypted there

Site	Ping duration	Jitter		Overall
		World's avg = 23ms (according to speedtest.net)	Tested broadband avg = 3.27ms (according to cloudflare.com)	
Stackoverflow.com	min/avg/m ax = 60/61/63ms			57.73 ms
Linkedin.com	min/avg/m ax = 17/20/19ms			16.73 ms

won't be any trend that would be depicted on the analysis thus one can't say if there is data hidden in the image.

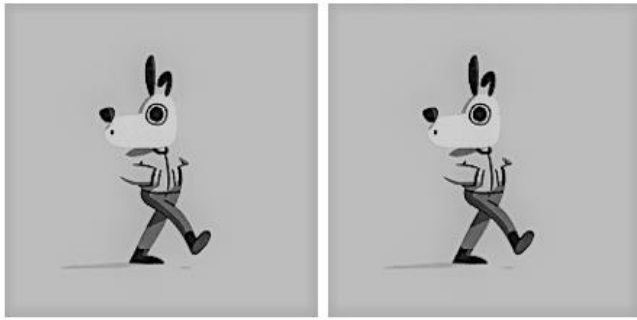


Figure 13 Sample image before and after encoding data

The possibility of an adversary at the middle of the connection reading the data is thus turned down to none. As we saw in the second phase, the MITM attack phase, the adversary never gets any data on the image just the reference to that image is provided to the adversary as ``, thus the detection and sniffing of the sensitive data on the network turns down to none, and even if the adversary manages to get hold of the image the output image after processing won't give any trace of data hidden in the same as there would be no trend depicted on steganalysis. Thus, to get any sign of data adversary must have the original image with them and even if so happens that they have it, they won't be having the private key to decrypt the data and reach the readable content.

IV. Computation Time

Computing all the steps are a separate topic but the comparison of time between the HTTPS/HTTP connectivity formation and the proposed protocol of HTTPPI might be a set off if it could not reach the same level, as the connection speed plays a great role during the attack and security phases and even based on the a longer computation time the adversary might be able to make out the image and data size along with the encryption key size which on longer time scale might have a negative impact thus comparison of the throughput time and the time connectivity formation including encryption and steganography has to be taken into account.

For the base values to be compared with the connectivity formation for the site being used for the test purpose were used again in order to provide with the ping time, to reach the host and back and the jitter time, to consider the time delay in sending the data packets over the connection used, whose combined result in connectivity formation by sharing data request and response packets.

In order to compare the proposed model timing with these values 5 test phases were carried out and average

of all the values has been considered for the comparison. As the real-world network delays and jitters can't be added on a simulation, the real-world values are added to the average values to carry out nonbiased

Table 1 Testing different websites

comparisons.

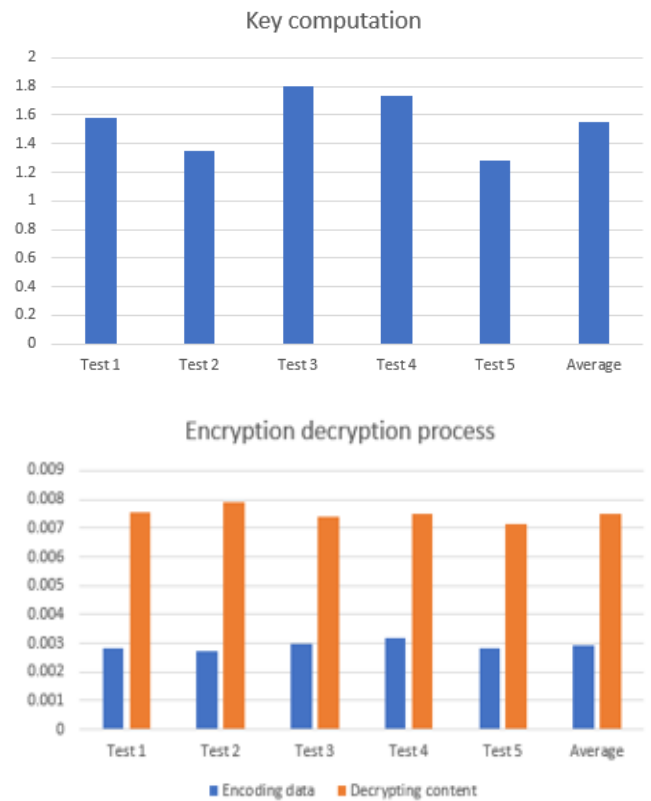


Chart 1 Key computation & data encryption-decryption time taken

As the keys can be computed and shared apart from the connection formation, thus not affecting the time in any sort.

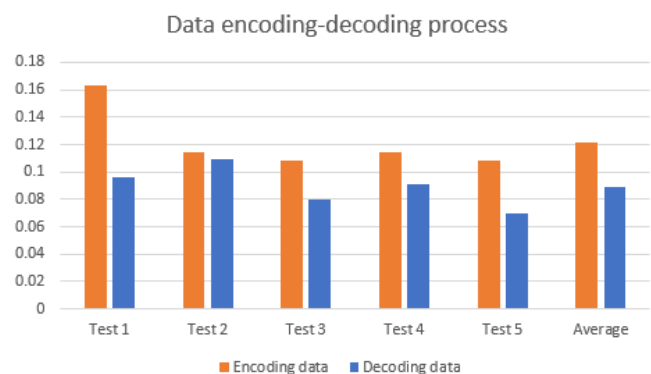


Chart 2 Data encoding and decoding execution time

Computing time for whole process together and adding the real world values for un biased comparison.

	Execution process	Time taken (ms)
Sender	Computation of encrypted data & encoding the encrypted data	0.124692
Receiver	Decoding the data from image & decrypting the data	0.965990
<i>After adding the delays and jitter (20.97ms)</i>		
Sender	Compute everything and send the data	21.094692
Receiver	Receive the data and reach the required information	21.93599
Overall required communication duration		43.030682

Table 2 HTTPPI execution and connectivity time

VIII. CONCLUSION

As per the results above and the output reached, the proposed model of image protected hypertext transfer protocol (HTTPPI) was able to provide more secure connection compared to HTTPS and HTTP, as even the adversary be able to perform MITM attack upon the HTTPS connection and sniff the data, the sensitive data would be encoded in the image which won't be appearing to the adversary as shown earlier. Even if the adversary gets their hand on the image they won't have the input image to compare the output image with in order to reach the required set of encrypted bit value. Even in case say the adversary get the original image from somewhere the data decoded from the image would be RSA-3072 encrypted, where without the secret private key of the target the adversary won't be able to break the encryption. The proposed model of HTTPPI performs well considering the execution and connectivity time faster than the few HTTPS protected websites while slower than few. This method can be considered for usage in cases where security is priority over speed.

FUTURE WORK

The HTTPPI protocol works as expected, still there is chance of improvement in the time taken for computation, a faster way can be found based on parallel computing designs and as we know RSA is a slow algorithm, thus a faster form of encryption can be considered for future use and development.

REFERENCES

- [1] Artz, D. (2001). Digital steganography: hiding data within data. IEEE Internet computing, 5(3), 75-80.
- [2] Subhedar, M. S., & Mankar, V. H. (2014). Current status and key issues in image steganography: A survey. Computer science review, 13, 95-113.
- [3] Eggers, J. J., Baeuml, R., & Girod, B. (2002, April). Communications approach to image steganography. In Security and watermarking of Multimedia Contents IV (Vol. 4675, pp. 26-37). International Society for Optics and Photonics.
- [4] Chandramouli, R., & Memon, N. (2001, October). Analysis of LSB based image steganography techniques. In Proceedings 2001 international conference on image processing (Cat. No. 01CH37205) (Vol. 3, pp. 1019-1022). IEEE.
- [5] Anderson, R. J., & Petitcolas, F. A. (1998). On the limits of steganography. IEEE Journal on selected areas in communications, 16(4), 474-481.
- [6] Provos, N., & Honeyman, P. (2003). Hide and seek: An introduction to steganography. IEEE security & privacy, 1(3), 32-44.
- [7] Kumar, D., Ma, Z., Durumeric, Z., Mirian, A., Mason, J., Halderman, J. A., & Bailey, M. (2017, April). Security challenges in an increasingly tangled web. In Proceedings of the 26th International Conference on World Wide Web (pp. 677-684).
- [8] Chomsiri, T. (2007, May). HTTPS hacking protection. In 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07) (Vol. 1, pp. 590-594). IEEE.
- [9] Kocher, P. C. (1996, August). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Annual International Cryptology Conference (pp. 104-113). Springer, Berlin, Heidelberg.
- [10] Callegati, F., Cerroni, W., & Ramilli, M. (2009). Man-in-the-Middle Attack to the HTTPS Protocol. IEEE Security & Privacy, 7(1), 78-81.
- [11] Subhedar, M. S., & Mankar, V. H. (2014). Current status and key issues in image steganography: A survey. Computer science review, 13, 95-113.
- [12] Fiore, D., & Gennaro, R. (2010, March). Making the Diffie-Hellman protocol identity-based. In Cryptographers' Track at the RSA Conference (pp. 165-178). Springer, Berlin, Heidelberg.
- [13] Nan Li, "Research on Diffie-Hellman key exchange protocol," 2010 2nd International Conference on Computer Engineering and Technology, 2010, pp. V4-634-V4-637, doi: 10.1109/ICET.2010.5485276.

- [14] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Third Edition, Pearson International Edition
- [15] Shawkat, S. A. (2007). Enhancing Steganography Techniques in Digital Images (Doctoral dissertation, Faculty of Computers and Information, Mansoura University Egypt-2016).