B. TECH PROJECT REPORT

On

Design and implementation of protocol for encryption of data and preventing replay attack in the network.

BY

ABHISHEK MISHRA

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

DECEMBER 2019

Design and implementation of protocol for encryption of data and preventing replay attack in the network

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE AND ENGINEERING

Submitted by :
ABHISHEK MISHRA

Guided by :
DR. NEMINATH HUBBALLI
Associate Professor
Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2019

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **Design and implementation of protocol for encryption of data and preventing replay attack in the network** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering completed under the supervision of **DR. NEMINATH HUBBALLI, Associate Professor , Department of Computer Science and Enginnering** IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

ABHISHEK MISHRA
B. Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

DR. NEMINATH HUBBALLI
Associate Professor
Department of Computer Science and Engineering

PREFACE

This report on **Design and implementation of protocol for encryption of data and preventing replay attack in the network** is prepared under the guidance of **Dr. Neminath Hubballi**. This report is the result of my internship at GE Healthcare, Bangalore. This report includes a brief description of the reason for encryption and replay attack prevention. It discusses various types of encryption algorithm. The implementation of these encryption algorithm is done to determine the most suitable algorithm. It includes the design for protocol to incorporate encryption and replay attack prevention into the existing application layer protocol. I have tried to the best of my ability to explain the content in a lucid manner. I have also added figures to make the report more illustrative and easily understandable.

ABHISHEK MISHRA
B. Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

ACKNOWLEDGEMENT

I would like to thank my B.Tech Project Supervisor **Dr. Neminath Hubballi** for his valuable guidance and support throughout the duration of the project and for giving me this opportunity to work in this interesting domain. Without your support, this report wouldn't have been possible

I would like to thank **Mr. Lokesh Aradhya**, who is Director at Monitoring Solutions Department at GE Healthcare. He provided constant guidance and helped me understand various technologies. Whenever i got stuck in some problem, his advice always proved useful.

I would also like to thank my friends and family who provided constant encouragement and enabled me to perform at best of my abilities. I am really grateful to Department of Computer Science and Engineering, IIT Indore for this opportunity. I offer my sincerest thanks to all the people who were directly or indirectly involved in this project

ABHISHEK MISHRA
B. Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

ABSTRACT

The GE Hospital Network has many monitoring and data-processing devices and earlier real-time data moved freely through the network in plain-text format. This made it very easy to steal critical patient information and also made the network vulnerable to cybersecurity threats.
In this project, I have worked on making protocol changes to encrypt the data before it enters the network. This involves determining the appropriate algorithm for encryption so that the time taken to encrypt doesn't put too much lag into data transfer. All the algorithms are theoretically unbreakable, it is dues to mishandling of keys and Initialization Vector that makes the encryption weak, the protocol developed ensures proper key management, key rotation and initialization vector generation so that the encryption doesn't have any vulnerability.
There are many threats that remain after encryption such as replay attacks on the network so this project includes building a mechanism to prevent replay attack by adding context information in the packet (in cases where time information is not accurate). After developing the protocol, it was implemented on 2 GE Applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

.

1.1 Network

Monitoring solution team makes devices that are present for monitoring patients inside the hospital such as bedside monitors and nurse workstation for monitoring multiple patients for continuous real-time monitoring of critical patient data such as heart rate and respiration rate. These devices include an application along with a computer system designed to facilitate the application.
All the GE devices in the Hospital network work on a common application layer protocol known as Unity protocol. Government regulations require that the patient data is to be stored according to certain guidelines as mentioned in HL7 standard. So there is one more device besides the monitoring devices which has a sole task of converting data from GE standard to HL7 standard so that it conforms to appropriate guidelines for data storage. This device handles the traffic from multiple other monitoring devices so functioning of this device is critical for the hospital to function.
All these devices are interconnected with a LAN inside the hospital and critical patient data moves from one device to another using a custom GE application layer protocol. This protocol is majorly based on UDP protocol. These applications are "plug and play" i.e they don't require any handshake as such before they start functioning.

1.2 Protocol description

1. Once the application starts, it creates a thread and starts emitting broadcast packets into the network to announce its presence.

2. It contains all the info that might be needed by other applications in the network to identify the type of application and sent appropriate requests to the application based on its functionality.

3. Once the broadcast packet is received the recipients determine the functionality of sender and send UDP unicast messages to carry request and response.

4. Since these applications stream out continuous rates of data so UDP works well as application can tolerate few packet drops.

1.3 Problems with the existing protocol

The critical patient data is sent in plain text i.e. without any encryption.

1. The data stolen can be misused. For ex- Targeted advertisement to sell fake medicines to desperate terminally ill patients.
2. There are certain regulations that must be followed. Failure to do so can lead to fines.
3. Patients have the right to privacy.
4. Sending data in plain text format can be used to gain information about the network which can be misused by hackers.
5. There is no authentication mechanism to ensure only valid entities can communicate with the application.

1.4 Objectives

The expectation from the new protocol is as follows:

1.4.1 real-time encryption

To determine which encryption algorithm will be suitable, the application transmit real time data so time delays due to encryption and decryption can cause a lag in the application which can cause major problems in patient monitoring.

1.4.2 protocol for facilitating encryption

To make application layer protocol changes to facilitate encryption such as key management, initialization vector etc.

1.4.3 protocol for prevention of replay attack

Now encryption of packets ensures data confidentiality but it is still vulnerable to a form of attack known as a **Replay Attack**. Replay attack might allow hackers to gain control to the system or even permanent damage to it without knowing what is inside the packet or breaking the encryption.

1.5 Explaining replay attacks

Replay attack is a form of attack in which a malicious entity snoops into a computer network and intentionally delays or re-sends packets which can cause the recipient to malfunction. [1]

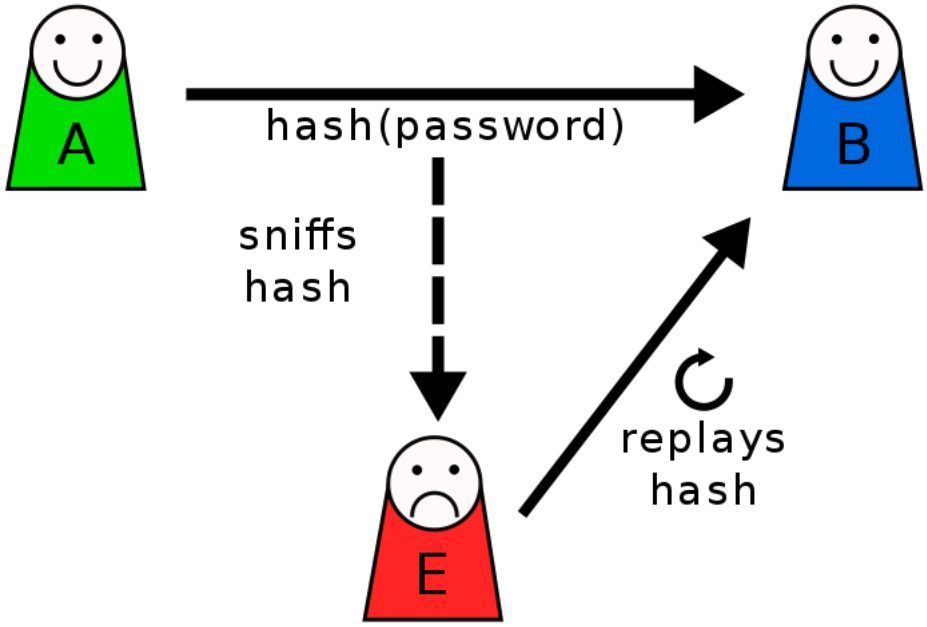

Figure 1.1: Illustration of a Replay Attack

[2]

In this case entity E can sniff out the packet containing the hashed password and resend it to authenticate itself as entity A and gain access to confidential information. The attacker doesn't need to know what is inside the packet or whether the data is encrypted or not as replaying an already valid packet ensures that it will be properly authenticated and decrypted by the receiver properly. So even after encryption, in order to prevent replay attacks on the network, there was a need to make changes in the protocol to prevent replay attacks.

Chapter 2

Determining the right Algorithm

2.1 Basic Constraints

The application requires that the encryption to be done at such a rate that 1 million byte arrays of 1000 bytes each be encrypted and decrypted in less than or equal to 20 secs on the dedicated hardware, If this rate is not maintained the packet queue size would keep on increasing as the packet processing rate is lower than the rate of arrival of incoming packets, leading to crashing of application.

2.2 Factors that determine the right choice of Algorithm

There are 2 main factors which governed the choice of algorithm.

- Most of the devices on the network are real time so encryption should be fast so that device performance is not affected.
- The protocol for communication between the devices is based on UDP, so one cannot rely on the packet dispatched to reach the endpoint.

2.3 Test Setup for constraint

To check whether an algorithm works or not, the idea was to make a small code using the choice of algorithm that would encrypt and decrypt 1 million packets.

There are 2 main paradigms for encryption:

- Asymmetric Encryption
- Symmetric Encryption

2.4 Asymmetric encryption

In Asymmetric encryption there are 2 kinds of keys, public Key and private Key.
As the name suggests, the public key is shared in the network and the private key is kept secret. Public key can be used to encrypt a message for the receiver which the receiver decodes using his private key.

Key management is convenient and key rotation is trivial as key is randomly generated every time the application starts. Once key exchange successfully happens a secure channel is created for every communicating device.

Incorporating asymmetric encryption involves adding a key exchange mechanism in the existing protocol but this is difficult in case of UDP protocol as UDP doesn't ensure that the packet dispatched would reach the target. Every device has to maintain a table for all other device and their corresponding public key.So to use Asymmetric encryption, a lot of changes need to be made in the existing application.
Asymmetric algorithms tend to be slower as compared to the symmetric ones since our application requires real-time data encryption where speed is a major factor, asymmetric algorithms rarely satisfy the constraint. For asymmetric encryption, I implemented RSA.

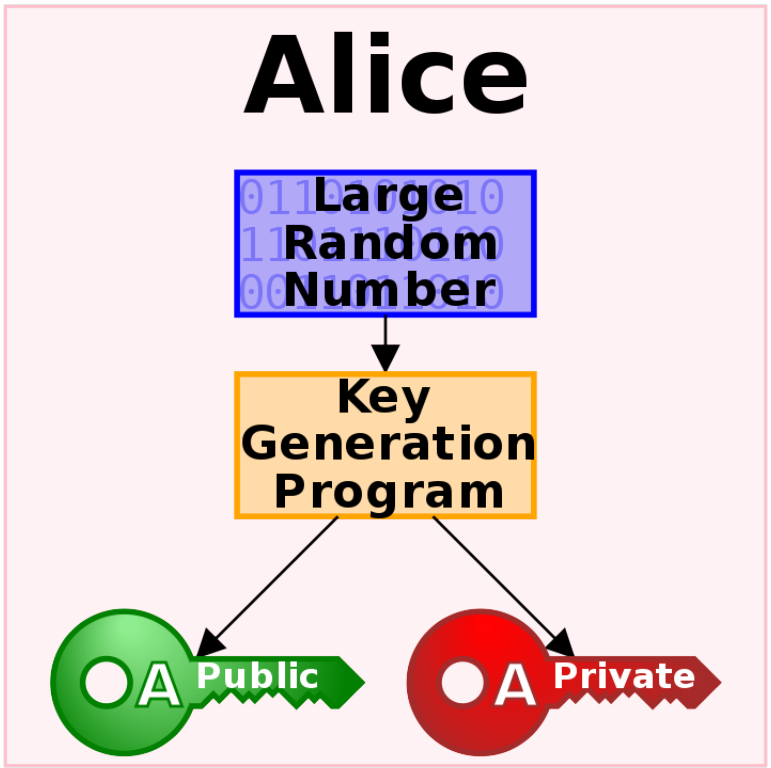

Figure 2.1: Key Generation in Asymmetric Encryption
[3]

2.4.1 RSA

RSA is one the oldest and most popular asymmetric key algorithm. The concept for this algorithm is that it is difficult to factorize large numbers. The math for this algorithm is based on modulo exponentiation. RSA is essentially a deterministic algorithm which means that same plain-text will form same cipher-text if same key is used. This makes it easier to break the encryption by pattern matching to decipher the contents.

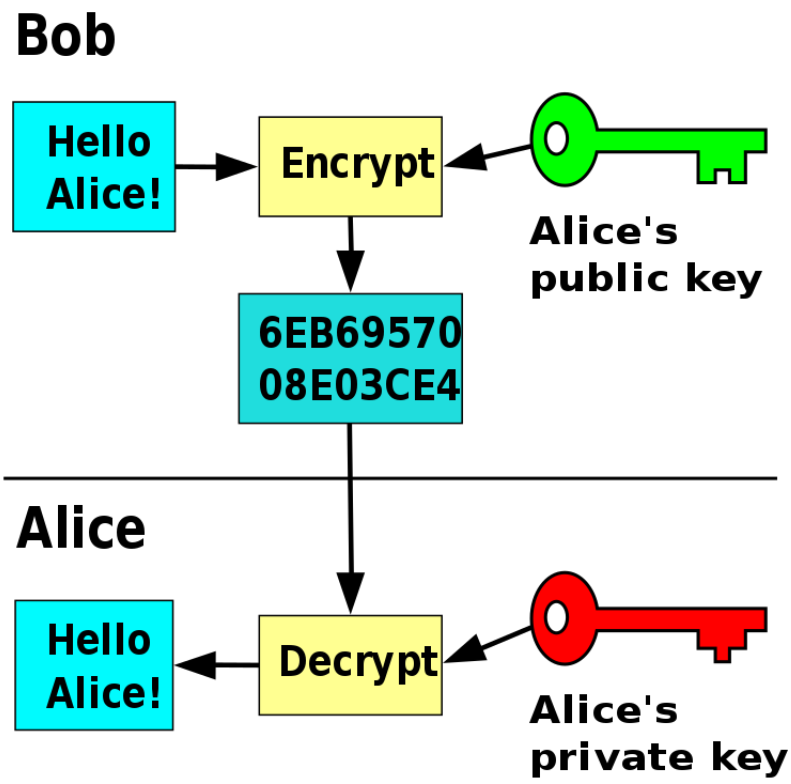

Figure 2.2: Illustration of public key encryption

[3]

Implementation: Made the java code using java cryptographic extension(JCE) as mentioned in the test setup for constraints

2.5 Symmetric Key Encryption

In symmetric cryptographic same key is used for encryption and decryption. Apart from a key, a second variable called nonce or initialization vector is used for encryption as well. For ex:- If S is the string to be encrypted, I is the initialization vector, K is the key.
Then encryption is enc(S,I,K)=S.enc And Decryption is dec(S.enc,I,K)=S
The same IV and nonce is used for both encryption and decryption. So the hacker has to know both key and initialization vector to decode the message. So Key is kept secret while Initialization vector is public.

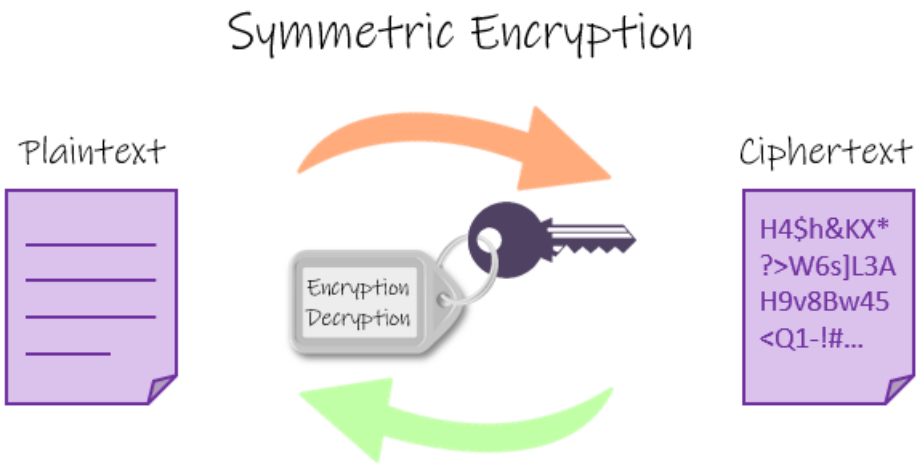

Figure 2.3: Illustration of symmetric encryption
[4]

Symmetric key algorithms tend to be faster and use less memory and CPU resources.Since the key and initialization vector as be same so both of them cannot be randomly generated in both server and client side. Key is generally hard-coded and initialization vector is embedded inside the packet. So key management and key rotation is a problem. For Symmetric encryption i tried the following algorithms:

- AES-CBC
- AES-GCM
- Xchacha20-Poly1305

2.5.1 AES-CBC

AES encryption happens over blocks of 16 bytes. A plain-text bigger than that is divided into blocks of 16 bytes to carry out rounds of encryption in each. In AES-CBC, each plain-text block is XORed with the ciphertext of the previous block. This ensures that a single letter of plain-text is changed, the change doesn't remain local to the corresponding ciphertext block. In Fact this change propagates over the entire ciphertext blocks coming after it. This adds pseudo-randomness to the output and makes it difficult to find patterns in the output.

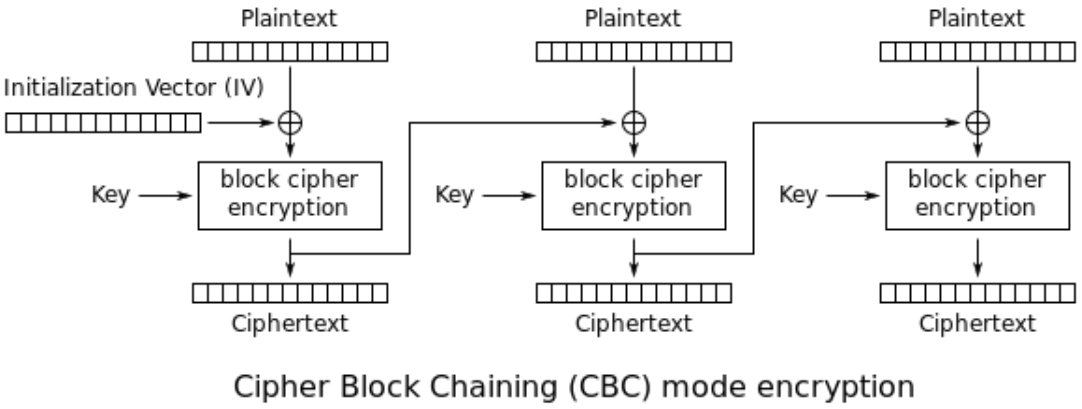

Figure 2.4: Illustration of AES-CBC encryption

[5]

Implementation: Made the java code using java cryptographic extension(JCE) as mentioned in the test setup for constraints.

2.5.2 AES-GCM

AES-GCM works like a stream cipher. This algorithm uses the counter mode of encryption together with galois mode for authentication. The design of this algorithm is in such a way that it can be parallelised better, this makes it run a lot faster as compared to AES CBC. It uses AES to generate pseudo-random number which is XORed with the plain-text to generate ciphertext. Like AES-CBC, AES-GCM also uses initialization vector to make the output non-deterministic w.r.t key. The initialization vector which is XORed with the first plain-text block adds a non deterministic nature to the relationship between key and plain-text.

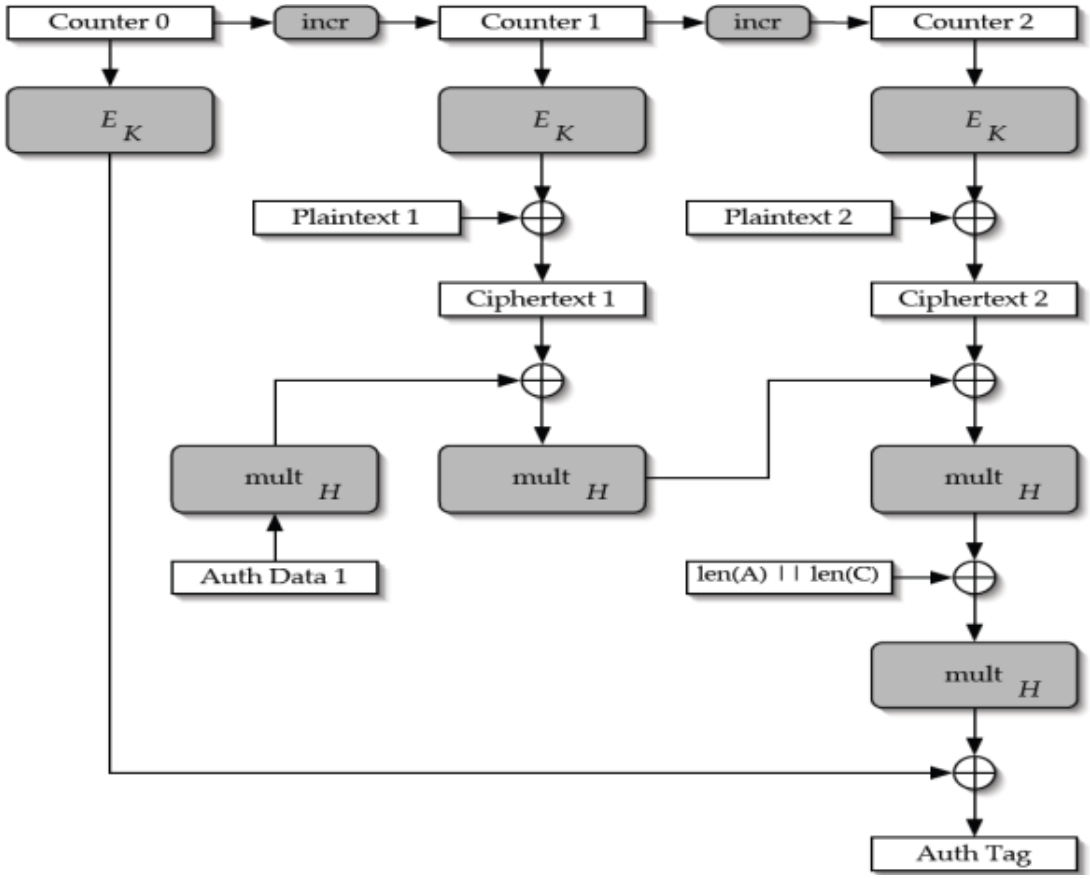

Figure 2.5: Illustration of AES-GCM encryption
[6]

Implementation: For AES-GCM wolfcrypt library is the best, but wolfcrypt isn't available for java so i wrote a JNI wrapper library to link wolfcrypt library from C language to Java.

2.5.3 Xchacha20-Poly1305

This is a modified version of the chacha20 cipher i.e. more number of bits for initialization vector. The cipher is a high speed one and performs better in software-only implementation as compared to AES. But the devices on the network have modern intel processors which has hardware support for AES so by testing only we can determine which would work faster.

Implementation: Made the java code using lazysodium JNI library which is built over libsodium written in C language as mentioned in the test setup for constraints. Stream ciphers tend to be faster in general.

2.6 Results

- **RSA:**Time taken was about 120-140 sec for the test setup which is not feasible as it doesn't satisfy the basic constraint.
- **AES-CBC:**Time taken was in the range of 4to 7 secs but it doesn't provide authentication system.
- **AES-GCM:**Time taken was 4-5 sec which is well within the accepted range. It also provides authentication feature.
- **Xchacha-Poly1305:**Time taken was 13-15 secs which is within the range. It provides authentication feature too.

Name of Algorithm	**Suitability**	**Reason (If no)**
RSA	NO	Constraint not satisfied.
AES-CBC	NO	Constraint satisfied but no authentication feature
AES-GCM	YES	
Xchacha20-Poly1305	YES	

Table 2.1: Comparison of Algorithms

2.6.1 AES-GCM (with hardware acceleration) vs Xchacha20-Poly1305

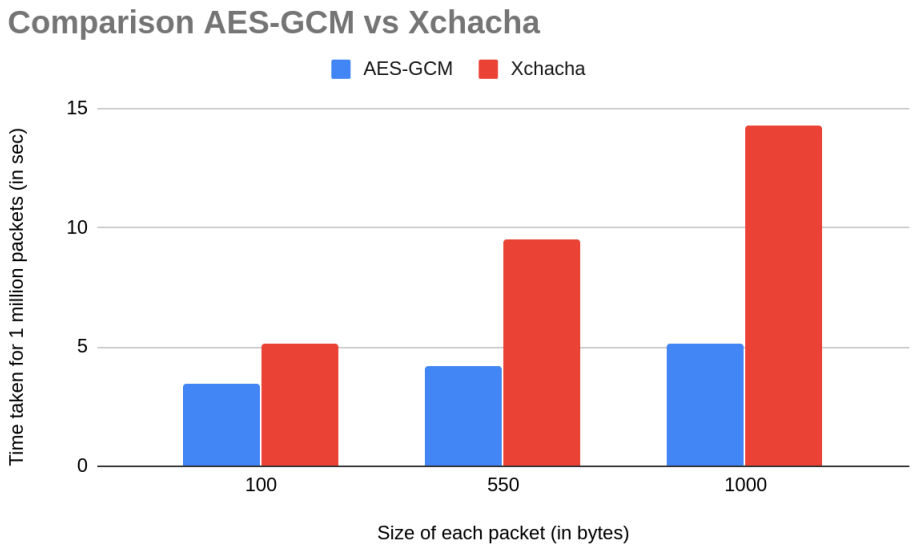

Figure 2.6: comparison of aes-gcm and Xchacha-Poly1305

Xchacha20 despite being a stream cipher which tend to be faster is working slower than AES-GCM. The reason for this is since AES was so commonly used, all modern intel processors have AES support built into the chip itself which makes an entire round of AES run fast. Wolfcrypt library has compiler options to enable this hardware acceleration if it detects the processor has AES support. This support for hardware acceleration was one of the major reasons to choose wolfcrypt library. So **AES-GCM is the best choice.**

Chapter 3

Protocol changes for enabling AES-GCM encryption

3.1 Initialization Vector and Key Management

Initialization vector is a 12 byte value is generated randomly for each packet and embedded inside the packet before sending the packet and this adds a non-deterministic nature to the output. In order to make the network invincible to pattern matching to decrypt packets, key-Initialization vector pair has to be unique for each message since initialization vector is randomly generated, after some time key has to be changed so that key-initialization vector pair doesn't repeat. This is handled using HKDF algorithm to generate new key from master key every time application starts. The same master key is hard-coded in all applications and 12 byte random salt value is

generated after every 60 minutes. The key for encryption is derived using HKDF key derivation algorithm using master key and salt. This key is used to encrypt the packet and this salt value is embedded in the packet. The receiver has the same master key. It extracts the salt value and

IV from the packet, uses same HKDF function to generate key and decrypts the message using the generated IV.

Since 12 byte is used for IV and 12 byte for salt, probability of pair repeating is:

$$1/2^{192}$$

which is negligible so key management problem is solved.

3.2 Authentication

AES-GCM algorithm has a built-in functionality for authentication. This authenticating data contains source IP, destination IP, source port number and destination port number embedded in the encrypted packet. The decryption algorithm is also given the expected authenticating data value, if this data doesn't match with the authenticating data extracted from packet then packet is discarded. The algorithm also produces addition 16 bytes of media authentication code. This can be used to verify the integrity of the data i.e. to ensure that data hasn't been modified intentionally or unintentionally.

3.3 Working of protocol

When the application starts, it'll generate a 12 byte salt which will be stored in a static variable. This variable will be updated every 60 minutes.This value is used to generate key via key derivation functions and a master key which will be hard-coded
When the application wants to send a message, it will use the generated key to encrypt the message along with an randomly generated 12 byte initialization vector. The Media Authentication Code generated, the salt value and initialization vector is put into the packet along with the encrypted payload.
The recipient extracts the salt value and uses it to generate the key required for decryption. Then it extracts the initialization vector. After that it decrypts the message and compares the Media Authentication Code extracted from the packet with the newly calculated media authentication code from encrypted payload, if they match and authentication is also successful then only data is forwarded for further processing.

Chapter 4

Protocol for prevention of replay attacks

4.1 Problem of replay attack

In order to prevent replay attacks, there must be a mechanism to identify the replayed or delayed packets. These packets are encrypted so one way is to put in some sort of context information in the packet. This value is put before encryption so cybercriminal cannot change or see or manipulate this value so if the packet has expired i.e it has outdated context value then hacker cannot do any tampering to correct the context value.

4.2 Time Stamp based context

One of the simplest ways to prevent replay attack is to simply add the UTC timestamp in the packet before encryption. Then based on network architecture, a critical time difference allowed is chosen. When the recipient receives a packet, it first decrypts it and checks the timestamp. If the time difference between the timestamp in the packet and current time is within critical time difference allowed then only the packet is valid.
If the packet is delayed or replayed, it reaches the target late and if the critical time difference allowed is small enough such packets won't reach the destination on time and will be rejected.

4.2.1 Problems with time stamp based approach

This mechanism only works if the time on all the devices are in-sync with each other. But it has been observed that due to time drift and other natural reasons, there is a time difference of upto 45 sec between devices so critical time difference allowed has to be greater than 45 sec. This is too long so the probability of replay attack successfully happening is high.

4.3 New protocol concept

The UTC time might differ in both the devices, but all devices are able to measure time elapsed quite accurately. So one device might have time X other device time Y but after 5sec devices will have X+5 and Y+5 time.
Before presenting the protocol, there are 2 keywords.
Base context: This is an integer value randomly generated for a session.
Current context: This integer value includes time considerations as well.

Current context= Base context + (current time - time at which base context was set).
So if device A knows that base context of device B is X. After 5 sec, device A will know the current context of device B is X+5.

4.4 Context exchange

As mentioned earlier, as per existing protocol when application starts it announces its presence in the network by broadcasting introductory packets which contain information about what the application is capable of doing. So now context exchange will happen using these packets.

- The application when it starts, generates a base context and starts broadcasting its base context to the network.
- There is a separate thread that listens to all the context from other devices in the network.
- On receiving base context value, this value is stored in a hash-table to calculate the current context value of other devices whenever it sends a request or response.

4.5 Packet structure changes

Now inside each packet a current context of the recipient is added and then the packet is encrypted. So devices guess each others context. If the context received is quite close to the predicted context then only the packet is considered valid otherwise it is rejected. So in each packet there is sender's current context and recipients current context.

4.6 Broadcast packet transfer

Device A on starting up chooses 1000 as base context.
Device B on starting up chooses 100 as base context.

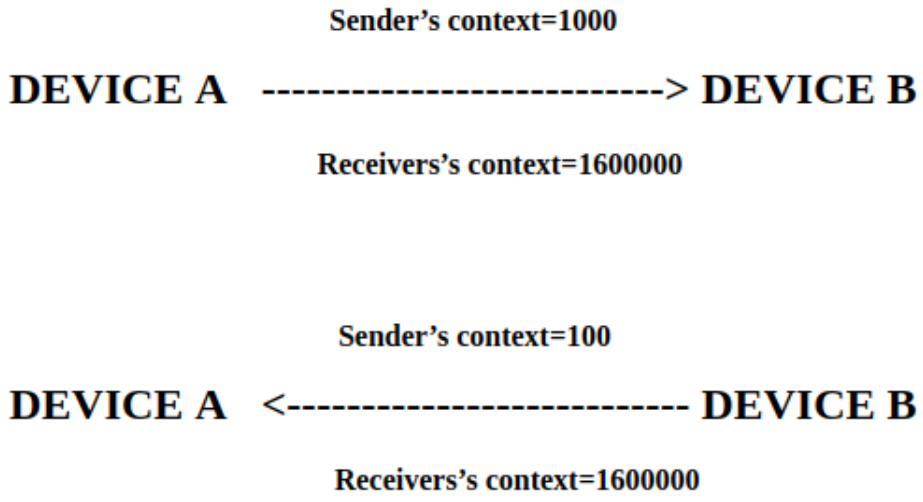

Figure 4.1: broadcast packet context

The broadcast packet has receiver's context set to UTC time as it doesn't know the receiver's context yet. This UTC time is used to verify the broadcast packet as broadcast packet doesn't carry any sensitive information so a weaker UTC based authentication can be used.

4.7 Unicast packet transfer

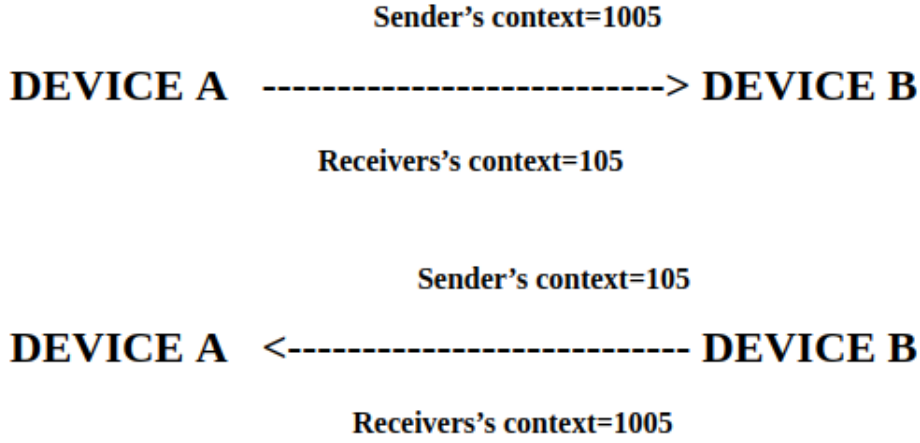

Figure 4.2: unicast packet context

So constraint imposed by the protocol is that each device tries to guess each other's context value. Device A on each packet puts a guess for device Bs current context. THis value is encrypted. Device B will accept only if it finds that guessed current context is reasonably close to the current context. Otherwise it discards the packet.

4.8 Calculation of critical time difference that can be allowed

The error allowed in guessing context should be short enough to give less time window for replay attack that's why unicast messages which carry critical data are checked for validity using context instead of UTC timestamp. Only broadcast messages are checked with UTC time as broadcast messages do not carry very sensitive info so a weaker check would work on them.
It should be large enough so that it is not affected by propagation and transmission delay. This number is decided based on the network structure.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The protocol developed was implemented on 2 GE Application device and it was observed that all the requirements were satisfied.
In this project, I have shown that choice of algorithm plays a very important role in ensuring that encryption happens faster. I have also shown a method to handle initialization vector generation and automatic key management using key derivation function which ensures that there are no patterns in cipher-text.
I have also shown how replay attack can be prevented in absence of accurate time in the device.

5.2 Future Work

These are the few issues on which work is going on:

- The master key is hard-coded in the code base. This makes the system vulnerable as the person having access to the code base can easily get the master key. So there is development going on for a way to dynamically assign master key by installing a key management server in the hospital. Once completed, this will eliminate the need to manually configure or rotate the master key.
- For accurately preventing replay attack, we need the time lag allowed for packet to be as less as possible so work is going on to dynamically derive this time lag value based on the network architecture and traffic.

Bibliography

[1] https://www.kaspersky.co.in/resource-center/definitions/replay-attack

[2] https://en.wikipedia.org/wiki/Replay_attack

[3] https://en.wikipedia.org/wiki/Public-key_cryptography

[4] https://www.101computing.net/symmetric-vs-asymmetric-encryption/

[5] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[6] https://en.wikipedia.org/wiki/Galois/Counter_Mode