KAUNAS UNIVERSITY OF TECHNOLOGY
ELECTRICALS AND ELECTRONICS FACULTY

AKASH RAVIKUMAR

EMBEDDED SECURITY OF IoT DEVICES

Master's Degree Final Project

Supervisor
prof. dr. ELIGIJIUS SAKALAUSKAS

KAUNAS 2016

KAUNAS UNIVERSITY OF TECHNOLOGY

ELECTRICAL AND ELECTRONICS FACULTY

EMBEDDED SECURITY OF IoT

Master's Degree Final Project

ELECTRONICS ENGINNERNG (621M61002)

Supervisor

prof. dr. ELIGIJUS SAKALAUSKAS

Reviewer

prof Doc. Paulius Kaskonas

Project made by

AKASH RAVIKUMAR

KAUNAS 2016

KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL AND ELECTRONICS

AKASH RAVIKUMAR

FINAL DEGREE PROJECT, PR00M118

"EMBEDDED SECURITY OF IoT DEVICES"

DECLARATION OF ACADEMIC INTEGRITY

10 June 2016

Kaunas

I confirm that the final project of mine, AKASH RAVIKUMAR, on the subject "EMBEDDED SECURITY OF IoT" is written completely by myself; all the provided data and research results are correct and have been obtained honestly. None of the parts of this thesis have been plagiarized from any printed, Internet-based or otherwise recorded sources. All direct and indirect quotations from external resources are indicated in the list of references. No monetary funds (unless required by law) have been paid to anyone for any contribution to this thesis.

I fully and completely understand that any discovery of any manifestations/case/facts of dishonesty inevitably results in me incurring a penalty according to the procedure(s) effective at Kaunas University of Technology.

(name and surname filled in by hand) *(signature)*

RAVIKUMAR, AKASH. Title of the project: EMBEDDED SECURITY OF IoT DEVICE. *Master's* Final Project Supervisor assoc. prof. ELIGIJUS SAKALAUSKAS, Faculty of Mathematics. Kaunas University of Technology.

Research field and area: EMBEDDED SYSTEMS SECURITY

Keywords: *Data recorder, cryptography, HMAC, signing and verification, IoT*

Kaunas, 2016. P 64

SUMMARY

In this master thesis the solution and microprocessor realization of cryptographic secure fiscal transportation data recorder is presented. Thesis solves the problems of transportation data authenticity and integrity. Data recorder is storing GPS coordinates, temperature values, time stamp and RFID reader during the transportation. The data stored in the recorder can’t be altered after a fraud attack. The symmetric cryptography methods were selected for this purpose. Therefore HMAC technique is used where a secret key was used along with a cryptographic hash function providing data authenticity and integrity. Even though HMAC may have advantages over other techniques, it do suffers from other attacks namely brute force attack and length extension attack. It is said that every cryptographic algorithm could be broken down based on the experience level of the attacker and based on the selected share key. Therefore in this research, HMAC technique was used to provide data integrity and authenticity.

RAVIKUMAR, AKASH. Projekto pavadinimas: ĮTERPTINIŲ IoT PRIETAISŲ APSAUGA. Magistro baigiamasis darbas/vadovas prof. ELIGIJUS SAKALAUSKAS, Matematikos fakultetas. Kauno Technologijos Universitetas.

Tyrimo sritis: ĮTERPINĖS APSAUGOS SISTEMOS

Raktiniai žodžiai: *Duomenų įrašymas, kriptografija, HMAC, prisijungimas ir verifikavimas, IoT*

Kaunas, 2016. 64 p.

SANTRAUKA

Šiame magistro darbe aptariamas kriptografijos apsaugotų įrašytų fiskalinių duomenų perdavimas mikrovaldikliu.Tezėje sptrendžiamos perduodamų duomenų autentifikavimo ir integralumo problemos. Įrašymo įrenginys saugo GPS koordinačių, temperatūros, laiko žymes ir RFID skaitytuvo duomenis, kurie yra perduodami. Šie saugomi duomenys negali pakisti neteisėtos atakos metu. Norint tai įgyvendinti naudojami simetriniai kriptografijos metodai. Taip pat naudojama HMAC technologija, kai reikalingas apsaugos raktas buvo naudojamas kartu su kriptografijos sumaišymo funkcija, užtikrinančia duomenų autentiškumą ir integralumą. Net ir geriausia HMAC technologija, lyginant su ktomis technologijomis, turėjo problemų kai buvo vykdomos brutalios ir ilgalaikės atakos. Sakoma, jog kiekvienas kriptografijos algoritmas gali būti nulaužtas priklausomai nuo įsilaužėlio ir naudojamo pagrindinio kodo. Taip pat HMAC technologija šiame darbe buvo naudojama duomenų integralumui ir autentifikavimui.

ACKNOWLEDGEMENTS

I would first like to thank **Kaunas University of Technology** for giving an opportunity to study master’s degree program, which I feel much important to me as, I not only studied about the subjects, but had to research on it and learnt something new every time, which in turn improved my knowledge..

I would like to thank my project supervisor **Assoc. Prof. Eligijus Sakalauskas**, **Mathematics Faculty,** for accepting me as his student to work on the project, which he handed on to me, not only made it look challenging, but made me more happy as I could finally achieve the needed result enhancing my knowledge.

I would like to express my deepest appreciation to **Prof. Vytautas Deksnys, Head of Embedded Systems Laboratory**, for being my technical supervisor throughout the course of my project phase. I would really appreciate him for being kind all the time, for responding to mail each and every time, for handing over the resources which was needed to complete my task and for his patience.

I would like to thank all the professors and associate professors, who handled lectures for me, as it was really helpful for me, to bring in the experience I acquired in the laboratory sessions handled by them and to apply it to my project work.

I would like to thank every member in my family for their love, support, sacrifice and motivation. Without them, I couldn’t have made it to where I am now. I thank them from the bottom of my heart. I wish to thank my father **Mr. B. Ravi Kumar, Retired Director, Electronics Testing and Development Centre and Centre for Reliability (ETDC & CFR), Government of India,** for constantly showering me knowledge, not just on my project work, but also for my lectures, laboratory works, examinations and whenever I needed him.

I would like to specially thank my friends **Mr. Inian Pavel, Embedded software Engineer, Prime Edge Info Solution** and my colleague **Mr. Aravind Prasad** for helping me with my technical issues of my project work from the beginning till end and provided me with tips, views and knowledge whenever I needed them. I also wish to thank all my other friends, who had my back, from the beginning till the very end of my study programme. I would like to once again give away my heartfelt thanks to every names mentioned here or I failed to mention here.

Contents

List of figures 8

List of tables 8

List of nomenclatures 9

1 Introduction 10

1.1 Aim of the research 11

1.2 Literature survey 12

1.2.1 Symmetric key cryptography 13

1.2.2 Asymmetric key cryptography 16

1.2.3 Threats to the cryptographic technology 18

1.3 Comparison between symmetric key cryptography and asymmetric key cryptography 20

2 Research methodology 25

2.1 Hardware development 25

2.1.1 System prototype 25

2.1.2 Project schematics and hardware development 26

2.1.3 Schematic and hardware description 27

2.2 Providing integrity and authenticity to data using hash message authentication code 27

2.2.1 Goals behind the construction of HMAC based system 28

2.2.2 Process flow at encryption phase 29

2.2.3 Definition of HMAC and implementation 29

2.2.3 Selection of key K 31

2.2.4 Truncated output 31

2.2.5 Verification of the message 32

2.3 Receiver Design 32

2.4 Experimental setup 34

3. Results and observations 37

4 Conclusion 46

References 47

Appendix 49

List of figures

Figure 1: Symmetric Key Encryption [1] 14
Figure 2: Public key cryptosystem [7] 16
Figure 3: Overall block schematic for the project 26
Figure 4: SHA-1 + HMAC Generation [22] 28
Figure 5: Process flow during encryption 29
Figure 6: Process flow of verification phase 32
Figure 7: Screenshot of the receiver system with password protection 33
Figure 8: Prototype and experimental setup of the recorder 34
Figure 9: Transmitting unencrypted data to the recipient 35
Figure 10: Data sent unencrypted to the password protected receiving system 35
Figure 11: Data sent along with MAC to the receiver 36
Figure 12: Data stored in a simple text file on the PC 37
Figure 13: reception of data illegally using terminal window 38
Figure 14: Screenshot of the password protected Receiver system 39
Figure 15: Terminal window of the attacker 41
Figure 16: Receiver system of the attacker with wrong secret key entered 42
Figure 17: Current consumption in low power mode 45

List of tables

Table 1: comparison between symmetric key cryptography and asymmetric key cryptography 20
Table 2: Cryptographic primitives and security goal 22
Table 3: Comparison of the devised cases 44

List of nomenclatures

GPS-	Global positioning system
GND-	Ground
USB-	Universal serial bus
UART-	Universal asynchronous receiver transmitter
RX PIN-	Receiver pin
TX PIN-	Transmitter pin
LQFP-	Low profile quad flat package
CRC-	Cyclic redundancy check
IDE-	Integrated development environment
GUI-	Graphical user interface
RFID-	Radio frequency identification
ADC-	Analog to digital converter
TTL-	Transistor to transistor logic

1 Introduction

When transporting sensitive materials and elements, proper care has to be taken during the transportation phase. Any elements that are sensitive to temperature must be taken serious care of this matter. In the drug transportation industry, there is a chance of the drug being damaged due to the rise in temperature as they are volatile in nature. But not just temperature will cause a damage to name of the transportation industry, but also the nature of the driver can decide the name for the company. If the driver has some split personality, where he might use the truck, not just for transportation of the intended goods, but he may also use the truck for transportation of other illegal substance. This could also spoil the name of the industry in the name of loyalty and integrity. In order to avoid this kind of exploitation of the truck during the transportation phase, there is a need to create a device that could not only provide the safety for the drugs to be transferred but also aid for the name of the transportation industry. But even if the device is created for the intended purpose, the data stored in the device can be tampered and can be manipulated. In order to avoid that from happening, the data must be stored safely. In this paper, it will be explained how the elements for creating the data recorder is selected and how the data inside the recorder can be secured.

The device will store different types of data to provide needed proof during the transportation of the materials. When this device is not connected to the network, it can be assumed it to be safe and secured. But if the system is connected to any network, it is prone to be attacked. In such case, all the data and information in that device will be compromised. In order to safe guard the device from being attacked or hacked, proper security must be provided to the system. In order to provide security, the technique of encryption should be employed. The most important deal of encryption is to convert the message into a cipher text, which is untranslated for a person. The cipher text cannot be understood normally by any person as the real message will not be present, but is converted. The key used for this process can be either symmetric key or a public key.

There is a term called Cryptanalysis, which is a study to obtain any needed information from the message which is in an encrypted form without the use of any additional information, such as a key. It can also be referred to as a study to hack any encryption algorithm and also their implementations. The person who carries out this process of Cryptanalysis is called as Cryptanalyst.

Therefore to keep our devices safe from cryptanalyst, we must know about what cryptography is, what its type are and how they can be attacked.

In this paper, there will be study made on all the encryption techniques based on its suitability to the problem. The best method will be chosen and will be compared to the other simpler method in order to provide security to the data's which are to be stored inside the data recorder. Since cryptography is a very wide field, only the best suited method will be discussed in the paper and the other techniques will only be used as the reference.

1.1 Aim of the research

1) To study the problems in transportation industry and to develop a device (portable data recorder) which can record huge amount data into its memory.
2) To study the existing techniques on how to provide integrity and Authenticity to the data which is being stored in the device.
3) Conduct experiments on whether the obtained data was properly encrypted and proper security was provided while sending and receiving and finally to view the data using graphical user interface only after being properly signed.

TASKS

A) To investigate power consumption in sleeping mode and when the interrupt is working during the low power mode at different time intervals (1 min, 5 min and 10 mins).
B) Measurement time according to the proposed data structure.
C) To estimate the sample rate.

1.2 Literature survey

As per many researches in recent years, there are more fraud cases seen in shipment industries and other types of transportation industries [1]. These cases do not seem to decrease but, it is increasing steadily and it occurs in many varieties as well. These fraudsters are being more and more ingenious in their modes of design and execution of schemes, including the use of modern technology to their advantage. These technologies can otherwise be referred to as computer hacking, but sometimes, there are some tried and tested 'old school methods, such as a document fraud might work just as well.

There are some types of fraud which are most common to happen and they are:

A) Bunkering fraud
B) Cargo and documents fraud
C) Chartering fraud
D) Cyber fraud
E) Information phishing

A) BUNKERING FRAUD

The fuel can be one of the greatest expense in the transportation industries. The fuel price has rose to an extent in last 10 years and hasn't taken a fall since then. Such values are said to be the stepping stone and acts as an incentive for criminal designs to commit fraud, the incidence of which is said to increase every day. The commons disputes and alleged misleading are in respect of quantity consumption of fuel, quantity of deliveries and quality of deliveries where supplier overstated quantity is supplied, trucks are adulterated and off specifications, the driver collude with other suppliers to short change on the supply of the fuel, trailers being invited to conduct illegal practises.

B) CARGO AND DOCUMENT FRAUD

These frauds can come in many different forms. It can involve all stale of cargoes that never existed, fraudulent misrepresentation of the documents, an attempt to illegally claim on the letter of credit, fake letters of indemnity and also other case like theft or cheating over quality and the quantity. This may also include forged bills of lading with the intention of stealing the trailer, a sub- charter of freight forwarded issues and the re-issues bills of lading with the supplies miss description and transporting a Trojan element which is not intended be legally supplied with other goods in the trailer.

C) CHARTERING FRAUDS

The main reason for this fraud is because of involvement of unknown or numerous intermediaries and an overly trusted representation by or on behalf of some first time counterparties. This type of frauds may involve the double play charter party, advance fee fraud and intercepted freight scam.

D) CYBER FRAUD

This is one of the fastest growing frauds that each and every companies face in this modern world. One of the key to make way for this is by having the information which are convincing enough and some knowledge to make the target believe that the transaction made was genuine. The information theft is therefore a key element to this fraud. The cyber-attack can elevate the wholesale theft of vast and confidential information.

E) INFORMATION PHISING

One of the key elements to the fraud is information. The fraudsters will seek to phish information from various sources, and it is known that in some companies, the criminal elements actively targets transporting agents as a source of information.

The frauds keep on increasing and to avoid such type of fraud, a key device must be designed and created to bring down the forgery in transportation and to ensure the safety to the goods to be transported. But the key thing is that, even if the recorder is created, the data stored in it must be secured and this can be done only with the help of proper cryptographic technique, or there will be purpose for the creation of this device. Since most of the suppliers who involve in forgery tend to tamper the information and produce a fake information. This if considered to be a federal offence in the United State of America. Certain Asian countries too have made strict rules for the supplier and the transportation to provide only reliable information.

Cryptography is classified into two types based on the keys used at encryption phase. They are:

1) Symmetric key cryptography
2) Public key cryptography

1.2.1 Symmetric key cryptography

In symmetric key cryptography, the sender and the receiver use the shared private keys. Where the sender uses a key, assumed as ‘KEY A”. This key will be used by the sender to encrypt the message and convert it to a cipher text. The cipher text is then decrypted at the receiver end using the receivers same shared key as given in Figure 1. The symmetric key ciphers where either implemented as block cipher or a stream cipher.

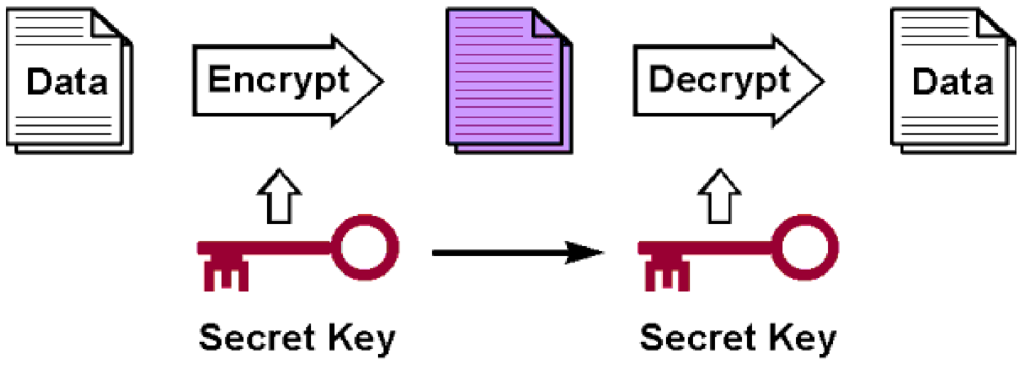


Figure 1: Symmetric Key Encryption [1]

There are many branches below symmetric key cryptosystems and public key cryptosystems. The symmetric key cryptosystems have the following techniques:

1) Data encryption system (DES)
2) Advanced Encryption System. (AES)
3) MD4
4) MD5
5) SHA-1
6) SHA-2
7) MAC

There may be 7 techniques which are given above for symmetric key cryptography, but not all of them are used in the embedded systems. The most commonly used techniques in Embedded systems are Advanced Encryption system (AES), Message Authentication code (MAC), Hash Message Authentication code (HMAC) and Data Encryption Code (DES). The other techniques also used for the systems, but they are not suitable for the research which will be carried out on the embedded system. The brief description are given about the other techniques.

A) DATA ENCRYPTION SYSTEM (DES)

DES is an algorithm that takes in fixed length string of plain text bits, and transforms it to through some series of complex operators. After this operations the plain text is converted to cipher text. The block size of DES is 64. It also uses a key for transformation, because of which, the decryption can be known by the concerned person, who only knows the key. The key might consist of only 64 bits but 56 bits key can also be used. Here, there are 8 bits, which are only used for parity checking and later discarded making the effective key length as 56 keys [2].

In this method the key is either stored or sent as 8 bytes and all of them with odd parity. It is said that, one bit in every 8-bit byte of the key may be used for detection of error while generating the key, distribution of the key or during storing. The bits 8, 16 and 64 are used for ensuring that each byte is of odd parity.

B) ASYMMETRIC ENCRYPTION TECHNIQUE (AES)

The AES is based on the principle of 'Substitution-permutation network'. Where it is the combination of both the permutation and substitution, and is considered to be the quickest algorithm in both the hardware and software. Unlike DES, the AES does not use any Feistel Network. AES is considered to be the variant of Rjindael, which consists of the fixed block size of 128 bits and its key sizes ranges from 128, 192 and 256 bits. By contrast, the specification specified with any of the block and key sizes that may be any multiple of 32bits, with both the bits of 128and maximum bits of 256. The AES operates on 4X4 column matrix of bytes, which is otherwise termed as state. It is although compared as to some of the version of Rjindael, which have larger block size and also additional columns in its state. A special field is needed to carry of most of the AES operations [3].

the keys used for the ciphers indicates the number of repetitions of transformation rounds that are used to convert the input, called the plaintext, into the final result, which is otherwise referred to as cipher text. The number of cycles are listed below:

1) 10 cycles of repetition for 128 bit keys
2) 12 cycles of repetition for 192 bit keys.
3) 14 cycles of repetition for 256 bit keys.

C) MD5 ALGORITHM

The Message Digest Algorithm a widely used cryptographic hash function which produces a 128 bit (16 Byte) hash value, which is expressed in a text format as a 32 bit hexadecimal number. This is mainly used for verification of data and is widely applied for many applications. The MD5 is considered to be a single was function, where both the encryption and encoding can be performed. The MD5 is most vulnerable to Brute-Force attack. This algorithm is the successor of MD4 algorithm. The security delivered by MD5 is not secure enough as the MD5 security is mostly compromised. Its weakness was clearly pinpointed and later exploited by

‘Flame Malware’ in 2012. This method is considered as “cryptographically broken and unsuitable for use” [4] [5].

D) HASH MESSAGE AUTHENTICATION CODE

In cryptography, the HMAC is a particular type of message authentication code, which includes the hash function in combination with a secret key. Any MAC can be used to verify both the data integrity and message authentication. Cryptographic function like MD5 or SHA1 can be used to estimate an HMAC. Such time of HMAC algorithm should be termed as HMAC-MD5 or HMAC-SHA1. The strength of the generated HMAC depends on the underlying strength of the function, the size of its hash output and on the quality and size of the selected key. An iterative hash function breaks up message into two blocks of fixed size and then they are iterated over the other with a compression function.

1.2.2 Asymmetric key cryptography

This is a type of cryptography technique where, sender and the receiver use two different keys for encryption and decryption. The public key will be used at the sender end for encryption and the private key will be used by the receiver of the information to decrypt the incoming message. Here the public key is computed from the secret keys and the public key can be distributed freely, whereas the private key must remain as the secret. The public key cryptosystem is shown in Figure 2 [6].

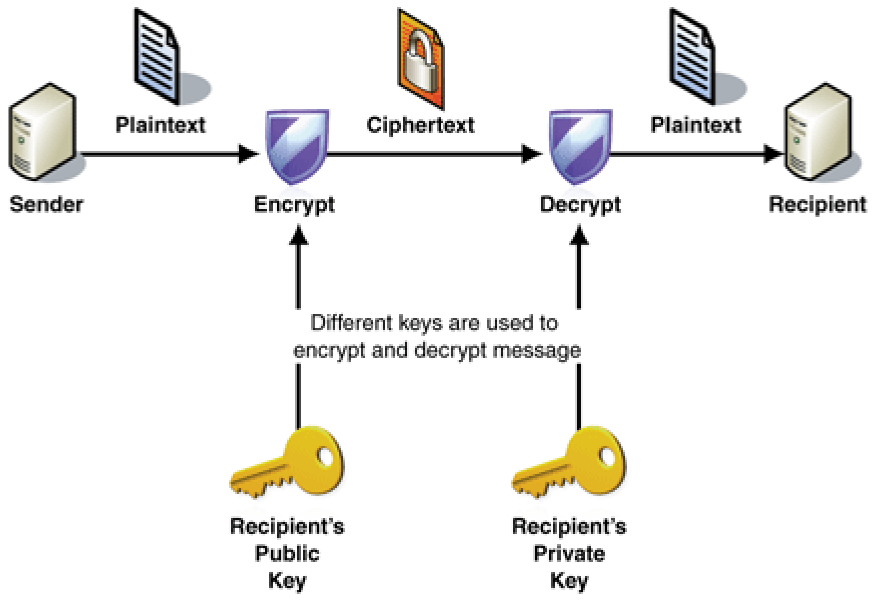


Figure 2: Public key cryptosystem *[7]*

There are some branch behind the public key cryptography and they are given below:

1) Diffie-Hellman key exchange
2) RSA Algorithm
3) Elliptic Curve cryptography (ECC)
4) ElGamal Encryption

A) DIFFIE-HELLMAN KEY EXCHANGE

The Diffie-Hellman is a way of generating shared secret between two people in such a way that the secret cannot be seen by observing the communication. To be clear, information is not being shared, but the key is created together. The basic steps behind the key exchange is:

1) User A and B comes up with two prime numbers g and p and informs user B
2) User A pick the secret number (a). The User A compute A= g^a mod p and send the result to user B.
3) The user B performs the same task by choosing secret integer (b) and later compute $B = g^b$ mod p and send this result to user A.
4) User A computes $S = B^a$ mod p
5) User B computes $S = A^b$ mod p

Now Both user A and user B have arrived at the same value S, because under mod p

$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$

$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$

In the above cases, the result for both the equation tends to be the same. Neither of the user's knows about others secret number, but the final result at both user ends will be the same number. The result obtained in step 4 and 5 will be the shared secret key.

B) RSA ALGORITHM

The RSA involves 4 major steps, they are, key generation, key distribution, encryption and decryption. It involves a public key and a private key. The public key can be known by everyone and it is used for encrypting the message. The private key is used for decrypting the message.

1) The first stage is to find three large value positive integers, e, d and n such that with modular exponential for all m:

$(m^e)^d = (\mathrm{mod}\ n)$

Even knowing e and n or even m, it will be difficult to determine d.

2) KEY DISTRIBUTION

   To enable the user B to send his encrypted message, the user A transmits pubic key (n, e) to user B through a reliable, but not a secret route. The private keys of these participants are always kept as a secret between them.

3) ENCRYPTION

   If user B wants to send a message M to user A, the message M is first converted into integer m, such that $0<m<n$ and $\gcd(m,n) = 1$ by using padding protocol. The user B then computes cipher text c, using user A public key 'e'.

   $C = m^e\ (\mathrm{mod}\ n)$

4) DECRYPTION

   The message is recovered by user A using his/her private key.

   $C^d = (m^e)^d = m\ (\mathrm{mod}\ n)$.

5) KEY GENERATION
   a) Two distinct prime numbers p and q are selected.
   b) Compute $n = pq$
   c) Compute $\phi(n) = \phi(p)*\phi(q) = (p-1)(q-1) = n - (P+q-1)$ where ϕ is Euler's totient function.
   d) Choose an integer e such that $1<e<\phi(n)$ and $\gcd(e, \phi(n)) =1$, where e and $\phi(n)$ are coprime.
   e) Determine d as $d=e^{-1}\ (\mathrm{mod}\ \phi(n))$

      It can be more clearly stated as $d.e = 1\ (\mathrm{mod}\ \phi(n))$.

      Where e is given as public key exponent and d is kept as the private key component.

1.2.3 Threats to the cryptographic technology

Even though we provide security to the systems using proper cryptography techniques, there are still attacks occurring on the systems. This is due to the simplicity of the key chosen or the computational level of the techniques algorithm [8].

The threats to the symmetric key encryption technique is given below:

1) Key search or Brute Force Attack
2) Cryptanalysis
3) System Based Attacks.

A) BRUTE FORCE ATTACK

The most common attack is Brute force attack. In this type of attack, the attacker simply tries all the possibilities and possible combination to find out the secret key. In this attack, most of the combinations may fail, but any one of the combination might succeed and lead to the possibility of hacking.

There is no way to defend against the key search attack. The reason is because the attacker will try to all the possible keys to decrypt a message. The key search is not very efficient. If the chosen key is large enough with numbers and symbols. It gets harder for the cryptanalyst.

B) CRYPTANALYSIS

Most encryption algorithms are said to be defeated using proper mathematics and computing skills. In this technique, many encrypted messages can be deciphered without knowing the actual key. A skill cryptanalyst can decipher all the encrypted data.

A cryptanalytic attack have two goals. Either the attacker may have a cipher text and discover a plain text from it or the cryptanalyst might have the cipher text and he/she wants to determine the encryption key, which was used to encrypt the message.

C) SYSTEM BASED ATTACKS

The other way to attack the system is by attacking the system which has the cryptographic algorithm directly. In this case, the attacker will not attack the algorithm directly.

The systems using asymmetric key encryption technique also suffer from various threats. The threats to systems using asymmetric encryption [9] is discussed below.

A) Man in the middle attack
B) Brute force attack

As the brute force attack is already discussed in early section. The other attack method will be discussed below.

A) MAN IN THE MIDDLE ATTACK

In this type of attack, a malicious 3rd party intercepts the public key, on its way to other members participating during the exchange of the keys. This malicious 3ed party can send his/her public key with his own message claiming to be from the intended sender. The attacker will keep on repeating this process until he can successfully impersonate each members, without the other parties knowledge by his deception.

1.3 Comparison between symmetric key cryptography and asymmetric key cryptography

S.NO	SYMMETRIC KEY ENCRYPTION	ASYMMETRIC KEY ENCRYPTION
1	Less computation steps	It has more computational steps
2	The encryption keys need not be very long.	The encryption keys should be usually longer
3	They vulnerable to brute force attack, but not as severe in case of public key cryptosystems	Keys in public key cryptosystems are more vulnerable to brute force attack
4	One key is shared for encryption and decryption	Public keys are shared between all entities
5	Algorithm is less complex and much faster	Algorithm is Complex and slower
6	It helps in bulk encryption of files and communication path	Key encryption and distribution of key

Table 1: comparison between symmetric key cryptography and asymmetric key cryptography

Therefore, from Table 1, though asymmetric encryption method is one of the best method. The choice is made for symmetric key encryption because of the computational speed, one encryption key for both purpose, bulk encryption of text and communication path.

Since an embedded system that only has to transmit the data, and does not have user controlling over it will be created during the research phase, there is no possibility for the embedded system compute a public key and transmit to the user, who uses the GUI.

Now that the technique of symmetric key encryption is selected. They type of symmetric key encryption must also be selected. The encryption only provides confidentiality to the message. The user encryption alone makes the message, which is encrypted vulnerable to a cipher text only attack. The sender with encryption key, will be able to encrypt the message as E(M). The valid information should be sent only by the holder of the key and no one else. But the attacker, who intercepts the cipher text can attack cipher text, alter it and make it say something else when decrypted. However, the larger the message and with bigger structure, the harder it becomes to be practically carried out.

But if a MAC is used along with encryption, the receiver will be able to compute the change in cipher text because the MAC will never compute. If the same key is used for Encryption and MAC. Then the sender can change the message too and encrypt it, and again send it as E(M). The hash is encrypted (our MAC) and the message is also encrypted (for confidentiality). By this way, it can be made computationally impossible to alter the cipher text and come out with a valid message, even if the sent message is single random byte. Choosing the proper hash function, the encryption and key length becomes better. In conclusion:

1) Encryption does not provide integrity itself.
2) MAC (Integrity) does not provide confidentiality by itself.

It is often better to combine cryptographic primitives to achieve many security properties.

There are different types of cryptographic primitives given in Table 2, which can distinguish the security goal which they fulfil, in simple, it can be referred to as a protocol of "appending to a message":

1) Integrity: it's the raise in the question whether the recipient is confident with the received message, on whether it is accidentally modified or not?
2) Authentication: will the receiver of the information be confident enough to determine whether the message has originated from the sender?
3) Non-Repudiation: if the recipient of the message has transferred the message to the third party. Can the third party be confident enough to trust, whether the message has originated from the sender?

CRYPTOGRAPHIC PRIMITIVES AND SECURITY GOAL	HASH	MAC	DIGITAL SIGNATURE
INTEGRITY	YES	YES	YES
AUTHENTICATION	NO	YES	YES
NON-REPUDIATION	NO	NO	YES
KIND OF KEYS	NONE	SYMMETRIC KEYS	ASYMMETRIC KEY

Table 2: Cryptographic primitives and security goal

As per some researchers, the Authentication without the confidence on the key which is being used is useless. For the digital signature, the recipient must be very confident that the verification code belongs to and is from the sender. In case of MAC's, the recipient must be very confident that the shared symmetric key is only shared with the sender.

An unkeyed hash of the message, if appended to the message, will only provide safety against the changes that are accidentally made to the message or hash itself, as the cryptanalyst hacking the message, can simply modify the message and can simply calculate the hash value and use it instead of the original one. So this process provides integrity.

If the hash is transmitted over a different channel, when it is protected, it can also provide protection to the message against any modification. This can be used sometime with hashes of very big file (for example, an ISO file) where the hash itself is being delivered in a channel over HTTPS, which this big file can be transmitted over an insecure channel.

The message authentication code, also known as the keyed hash, provide protection to the message against forgery by anyone who doesn't own secret key. This can mean that the receiver can forge any message, thus we can consider both the integrity and authentication (as long as the receiver does not have split personality), but this message isn't non-repudiation.

The MAC's can be originated from unkeyed hash like HMAC or it can be created directly as a message authentication code algorithms.

A digital signature can be create and it needs private key to be created. This has to be verified with the public key of asymmetric key pair. Again, only the person who owns the private key can create a signature. Normally, not everyone knowing the public key can verify it. Therefore, this provides all the three cryptographic primitives. The main disadvantage of the digital signal is that, it does not prevent from replay attack.

Mostly, all the signatures are generated from hash functions and they are often considered to be slower than the MAC functions, such as used normally when there is no shared key, or the non-repudiation property is important.

Based on the primitives and other information's discussed in the above section, HMAC will be selected over digital signal because of the speed of digital signature and its vulnerability replay attack. The other reason for not going only with some encryption method is also clear that the encryption alone would make the data vulnerable to cipher text only attack.

HMAC functions are of two types:

A) AES 128 CBC mode HMAC with shared Key K
B) Other standard HMAC functions like HMAC-SHA1, HMAC-SHA 256, HMAC-MD5.

The AES CBC [10] mode specifies algorithm for Additional Authenticated Encryption based on the combination of Advanced Encryption System (AES) in Cipher Block Chaining mode (CBC) of operation for encryption mode and a HMAC-SHA1 message authentication code (MAC). It is form of an authenticated encryption where in addition to providing confidentiality for the plaintext that is encrypted, but also provide way to checks its authenticity and integrity of the provided data. The algorithm is considered stateless and randomised. The algorithm is based on generic composition of common block chaining with HMAC, with encrypt then MAC method. It uses either SHA-1 or SHA256 algorithm to provide message authentication. Here the input key is 36 octet long.

During encryption, the AES CBC initialisation vector should be 16 octet long. Here, prior to encryption, the plain text is padded by appending a single '1' bit to the resulting strings as a few '0' bits are necessary to make the number of bits to 128. If the bits in the payload data is multiple of 128, then additional 128 bit padding is done. The plaintext is then encrypted using AES 128 in CBC mode using the encryption key, which is 36 octet long.

The message authentication code generated uses HMAC-SHA1. The key length of MAC is 20 octet. But the final result of HMAC-SHA1 is truncated to 16 octets, by removing the last 4 octets.

After the CBC decryption, the padding is being removed and this is done by deleting the final 1 bit and all the following ‘0’ bits. The data which remains finally are used to form the plain text. The validation process consists of computing the HMAC of the message. Then later truncating it to 16 octet by again removing the last 4 octets and finally comparing the MAC values. If the values are not found to be equal, they are rejected. Else the message is accepted.

The AES 128 CBC HMAC is not preferred for the recorder and instead standard HMAC function is used because of the following reasons:

1) Computation complexity, where need for blocks arise with initialisation vectors.
2) It has restrictions for key length.
3) It uses separate keys for encrypting and message authentication.
4) More operation and leads to more power consumption.
5) The initialisation vector must be added to the cipher text at each blocks making the operation flow more complex.

2 Research methodology

The research was first concentrated on the problems to create the device, whose stored data needs to be secured from the third party or so called frauds. Since the research says that 'temperature' is the factor to which more priority has to be given because, most of the transported materials are temperature sensitive. In order to maintain a steady temperature inside the trailer or to know if there is an increase in temperature, a temperature sensor can be added to the system. An additional upgrade can be given for better safety of the materials being transported. The refrigeration might be good close to the refrigeration unit, but it doesn't remain the same far away from the unit. So it is better to give more priority to the materials which are placed away from the refrigeration unit. In order to make sure the temperature is same even at the end, an RFID reader can be placed near the final box and it can read the tag status of the box at that current location [11].

In order to prevent 'Bunkering fraud', A GPS module can be fitted into the device and the coordinates can be recorded. This can prevent the driver from diverting from the course of his journey. If the driver indulges in the bunkering fraud, the transportation company will get to know about this incident with the data being stored to the device [12].

In order to avoid 'Bunkering fraud' again and 'charted fraud' which could lead in interception of the goods during transportation, the time stamp can be stored to the device. With the help of time stamp, the head of the industry will get to know, how long the truck stopped at once place, why did it stop and also will play huge role to calculate the journey time. This time stamp can either be generated from the microcontroller unit which is used in this device or GPS can be used to obtain the time stamp.

Finally, in order to avoid all the frauds which were discussed in above paras plus other frauds like 'cyber fraud' and 'information phishing', the stored in the MCU memory has to be encrypted or there is not purpose for the existence of the device. Since the data is made to be transferred through Bluetooth, the encryption of data is strongly needed.

2.1 Hardware development

2.1.1 System prototype

The system prototype is described in Figure 3

1) A microcontroller communicates with all the external devices through UART interface and temperature sensor alone communicates through ADC channel with the microcontroller.

2) Lithium Ion battery with nominal voltage of 5V, 2.6 Ah.
3) UART/TTL communications protocol converter

2.1.2 Project schematics and hardware development

The Figure 3 shows the overall block diagram for the project. The portable data recorder was constructed in order to test whether the selected methodology was enough to provide the much needed data integrity and authenticity. All the needed data's were obtained from the interfaces of the device and were converted in to string or a packet. Once this string is constructed, the shared key along with the hash is used to encrypt the data and the data must be stored in the device memory. Once the data is encrypted, it is now ready to be transmitted to the receiver over Bluetooth. As the data is being converted in Bluetooth, there will be a third party, who will be watching into the data which is being transmitted. Courtesy of encryption and hashing, the sent text appears as a cipher text to him. In general encryption, if changes are made to the cipher text, the receiver of the information will not be able to identify it. But when cryptographic hash function along with the message authentication code is used, the alteration in cipher text can be observed. The data is then been received by the actual receiver of the information, the receiver of the information then had to sign it to verify the message and only after the verification of the data, it can be viewed through a PC. The conclusions where made based on different approaches techniques used for attacking, where, there were three types of experiments carried out to check the data integrity and authenticity provided to the data by the device. In the first case, no encryption was done while the data was being transferred. In the second case, a simple password based GUI was used to view the data which was again sent unencrypted and in the third case, the encrypted data was sent to check the data authenticity and integrity.

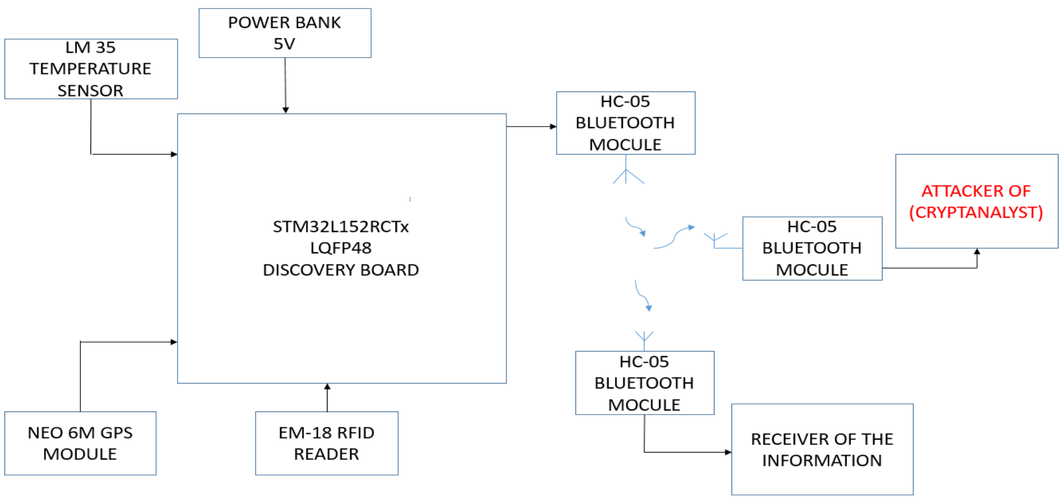


Figure 3: Overall block schematic for the project

2.1.3 Schematic and hardware description

The schematics and connection between the peripheral and the microcontroller of portable data recorder is given in the appendix. The Neo 6M GPS module is interfaced with the data recorder through a UART 1 interface. This GPS module is capable of operating at the voltage of 3.3V. The EM18 RFID reader is interfaced with the microcontroller through UART 3 interface. The reader operates at 125 kHz, with the supply voltage of 5V.the RFID reader has the range of 10 cm as the read distance. Since the UART only reads the tag status here, the RX pin is only connected to the microcontroller. The LM 35 temperature sensor is connected to the microcontroller with ADC line. This is because the used temperature sensor is an analog device and this pin is selected for that purpose. The sensor has the operation range from -50 C to 150 C and has the accuracy of 0.5 % at 25 C. The VDD and he DQ pin of the temperature sensor is connected with a pull-up resistor. The Bluetooth module is connected to the microcontroller through UART 2 interface. The communication range for this sensor is 10 meters radius, operates with low voltage range starting from 1.8 V to 3.3V, has the sensitivity of -80 dBm and it consists of an integrated antenna. All the devices interfaced with the microcontroller communicates with the baud rate of 9600bps, with one stop bits and with none parity mode. The microcontroller has the RAM memory of 256 KB and this is where the block of data's will be programmed to be stored after the encryption process.

2.2 Providing integrity and authenticity to data using hash message authentication code

To provide data integrity and authenticity is by using Hash Message Authentication Code, in short HMAC. The HMAC construction is said to make a cryptographic algorithm into a keyed hash. It is often used as the integrity protection, when the sender and receiver share a secret key [13]. The HMAC is a specific type of message authentication code involving with a cryptographic hash function (H') along with the combination of secret cryptographic key. Any MAC can be simultaneously used to verify both the data integrity and the authentication of the delivered message. Any cryptographic function such as MD-5 or SHA-1 can be used for the calculation of a HMAC, will be termed as HMAC-MD-5 or HMAC-SHA-1. The strength of the cryptographic hash function depends upon the cryptographic strength of the underlying hash function, the quality and the size of the key and also the size of the hash output [14].

The size of the Iterative hash function later breaks up a message into the block of fixed sizes and then iterates over them with a compression function. The MD5 and SHA-1 operates with the 512 bit blocks. The size of the output generated by a HMAC is the same as the underlying hash function. The size of the bits can be 128 or 160 bits in the case of MD5 or SHA-1, although they can be truncated to the desired size. The diagram of SHA-1 + HMAC Generation in shown in Figure 4.

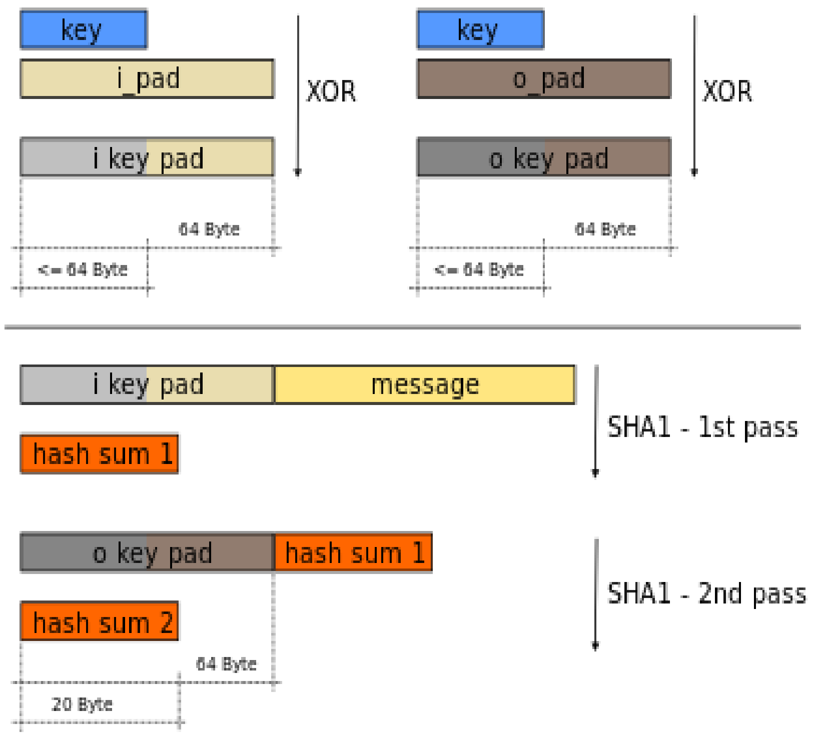


Figure 4: SHA-1 + HMAC Generation [22]

2.2.1 Goals behind the construction of HMAC based system

1) To use the function without any modification available with hash function. In general, the hash function operates well in the software [15].

2) It is needed to preserve the original performance of the hash function from without incurring any important degradation.

3) It is meant to use and handle the keys in a very simple and common way.

4) Based on the reasonable assumption underlying hash function, cryptographic analysis strength can be analysed

5) If faster or more secure hash function is needed, it should allow for easy replaceability of the underlying hash function.

2.2.2 Process flow at encryption phase

The process flow in the data recorder is given in **Error! Reference source not found.**, the primary objective is to obtain data into the recorder and then to convert it to a packet or a string, to make it easy for transmission purpose. Before the transmission phase, the data is encrypted with a shared symmetric key along with a cryptographic hash function. This data is then stored into the memory of the device. Generally, the data can be checked whether they are properly encrypted or not. If they are not encrypted properly, these data's can be encrypted again. The cipher text will be sent to the receiver and this message has to be signed to be verified. In the next section, the discussion of how has function is used along with symmetric key.

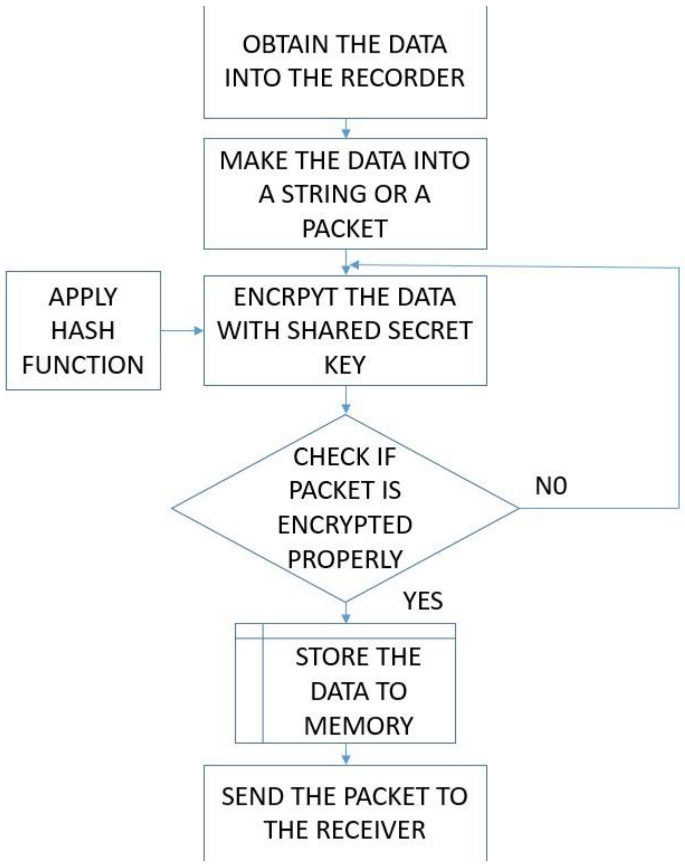

OBTAIN THE DATA INTO THE RECORDER
MAKE THE DATA INTO A STRING OR A PACKET
APPLY HASH FUNCTION
ENCRPYT THE DATA WITH SHARED SECRET KEY
CHECK IF PACKET IS ENCRYPTED PROPERLY
N0
YES
STORE THE DATA TO MEMORY
SEND THE PACKET TO THE RECEIVER

Figure 5: Process flow during encryption

2.2.3 Definition of HMAC and implementation

By the definition, the cryptographic Hash function is needed for the HMAC, which is usually represented by H and a secret key K. the assumption is made on H as a cryptographic Hash function, where the data is hashed by iterating a basic compression function on blocks of the data. The byte-length of a cryptographic hash function is denoted by B for blocks and the byte length of hash output is denoted by L.

The authentication key K, can be part of any length up to B, the block length of the hash function. The applications that uses more than B bytes will first hash their key K using H and then use the resultant L byte string as the actual key to HMAC. In any of the cases, the minimum recommended length of K is L bytes.

HMAC CODE:

The researchers demonstrated the HMAC with the following pseudocode. The block size selected for this purpose was 64 bits block. The pseudocode [16] is given below:

```
function hmac (key, message)
      If(length(key))>blocksize)
            Key = hash(key)
       end if
    if (length(key) < blocksize)
            key = key || [0x00 * (blocksize – length(key))]
    end if
        o_key_pad = [0x5C * Blocksize] EOR key
         i_key_pad = [0x36 * block size] EOR key
        return hash(0_key_pad || hash(i_key_pad || message))
           return function
```

In this method, there are two fixed and different strings used. These strings are named as opad and ipad. The ‘I’ and ‘O’ are the mnemonics used for the terms outer and inner.

ipad = the byte 0x36 is repeated by B times.

opad = the bytes 0x5C is repeated by B times

To compute a HMAC over a data ‘text’ the operation we must perform is given below:

H(K XOR opad, H(K XOR ipad, text))

Where:

1) Append zeros to the end of k to create the B byte string.
2) XOR (bitwise OR) the B byte string computed in the first step with ipad.
3) Then the stream of data ‘text’ is appended to the byte string B resulting from step 2.
4) The value of H is applied to the stream generated in step 3.
5) A bitwise or (XOR) is applied on B byte string computed in step 1 with opad.
6) The value of H is appended from step 4 to the byte string resulting from step 5.
7) Applying the H to the stream generated in the step (6) and the output to the result.

2.2.3 Selection of key K

The selected keys for HMAC can be of any length. The keys longer than B bytes are generally first hashed using H. But, less than L bytes is strongly discouraged as it would decrease the security strength of the function. Keys generally longer than L bytes are acceptable but for the extra length, it will not be significantly increased with the function strength. Any longer key may be advisable if the randomness of the key is considered to be weak.

The keys need to be chosen randomly, using a cryptographically powerful pseudo-code generator. Provided with random seeds. They should be periodically refreshed. The current attacks will not indicate a specific recommended frequencies for key changes s these attacks are practically infeasible. However, the periodic key refreshment technique is a fundamental security practise that aid’s in potential weakness of the function and keys, and limits the damage of any exposed keys.

2.2.4 Truncated output

A well-known practise with the message authentication code is on how to truncate the output of MAC & the output only part of the bits. The researchers Preneel and Van Oorschot [16] demonstrated some analytical advantages of truncating the output of the Based-based functions. The resolution that area wasn’t absolute as for the entire security advantages of truncation. The truncation has 2 advantages, where less information about the hash was available for the attacker and had a disadvantage where there were only less bits for the attacker to predict.

The application of the HMAC can be chosen to truncate the output of the HMAC by outputting the t leftmost bits of the HMAC, for the values of parameter t. The researchers recommended that the output length t to be not less than half of the hash output and also not less than eighty bits. They propose denoting a realization of HMAC that uses a Hash function H with t bits of output as HMAC-H-. The HMAC-SHA2-80 denotes the HMAC computed using the SHA-a function with the output truncated to 80 bits.

2.2.5 Verification of the message

The verification process is given in **Error! Reference source not found.**. The cipher text is obtained. Here the message is obtained along with a MAC. The hash will be computed for the received message along with the shared symmetric key. After computing the hash function for the received message, the system compares whether the computed hash equals to the received hash. If and only if the condition satisfies, the received message will be validated and considered to be from the authentic sender, else the message will be considered invalid and discarded.

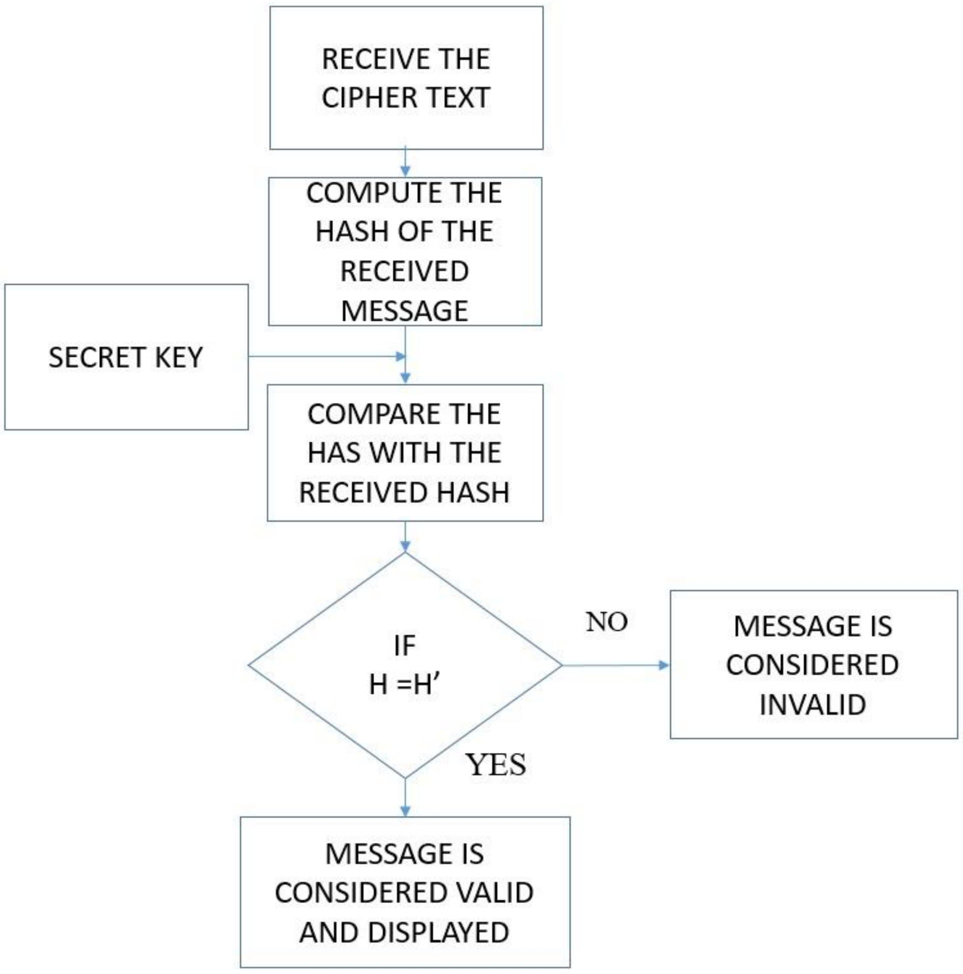

RECEIVE THE CIPHER TEXT
COMPUTE THE HASH OF THE RECEIVED MESSAGE
SECRET KEY
COMPARE THE HAS WITH THE RECEIVED HASH
IF H =H'
NO
MESSAGE IS CONSIDERED INVALID
YES
MESSAGE IS CONSIDERED VALID AND DISPLAYED

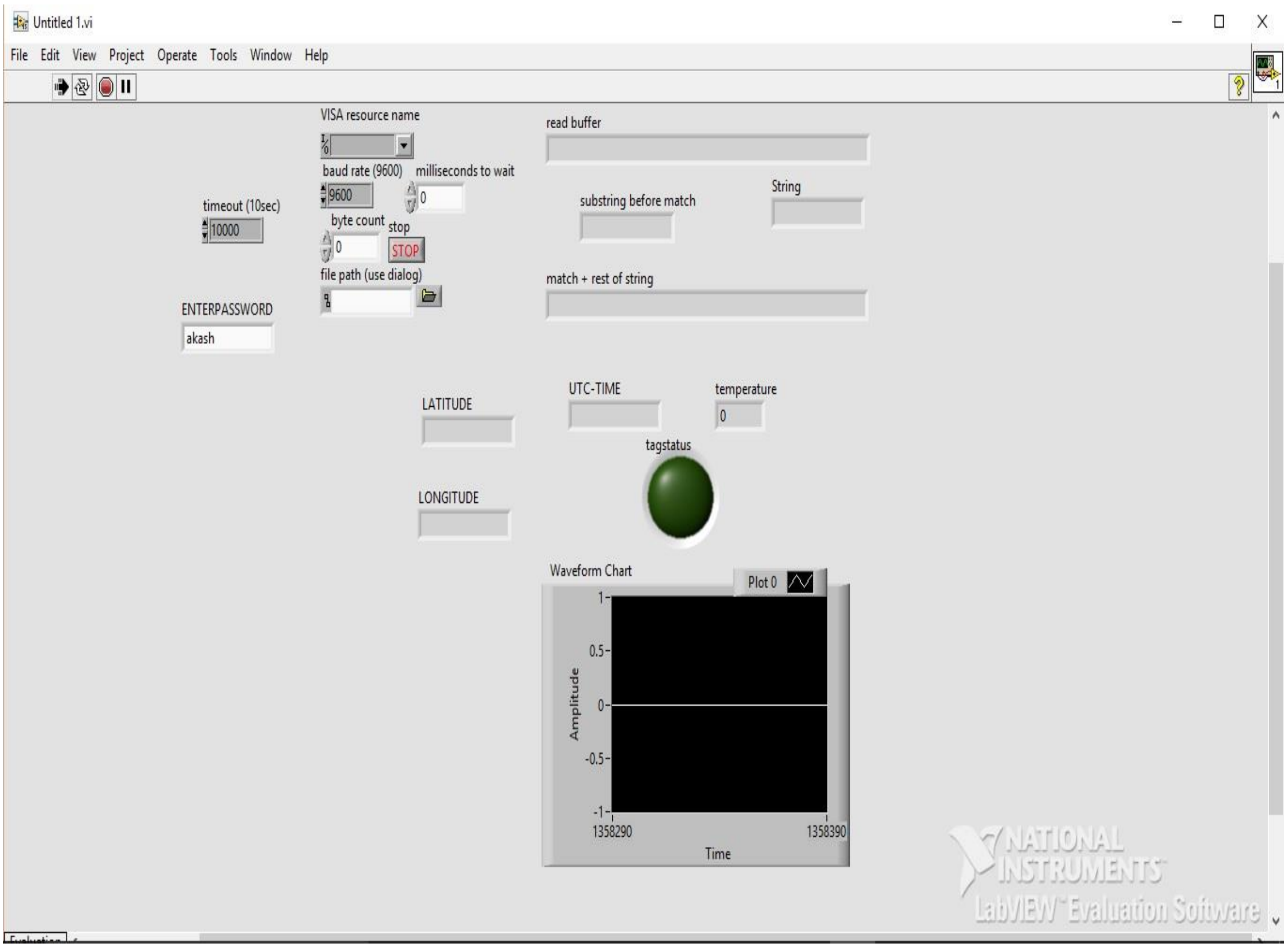
Figure 6: Process flow of verification phase

2.3 Receiver Design

The receiver for the device is programmed using LabVIEW. Initially com ports are to be initialised for this process. The incoming data will be received as a string. This string is made to be viewed in the user interface. The baud rate can be set in the receiver based on the device's communication speed. A timeout option is used to stop the process of receiving if there is no data to be received. Byte count option is used to receive only that amount of byte. This can be selected, if the length of the incoming message is known.

A file path dialog box is used. A file can be selected, so that the received data can be stored into the file. In password based system and for the secret key based system, a special dialog box will be used for that purpose. It is programmed in such a way that, only if the entered password is correct, the receiver starts to obtain data. Else, the data will not be received until then. Once the data is obtained. The data is then split, the latitude, longitude, UTC time and temperature are viewed through a dialog box specially allocated for them. In addition to that, the temperature will also be viewed through a graph. The RFID tag status is indicated through the LED. If the tag is detected, the LED glows, else the LED stays off.

In the case of receiver with verification mechanism, a secret key must be entered in a dialog box. The received message will be used along with the secret key to compute another hash function at the receiver end h'. This hash function is compared with the other hash 'h' received along with message. Only if the hash values are equal, the receiver allows the message into the system. Else, the receiver verifies the message as invalid and discards the message. The figure 7 shows the screenshot of the receiver with password protection.



VISA resource name
baud rate (9600): 9600
milliseconds to wait: 0
timeout (10sec): 10000
byte count: 0
stop
file path (use dialog)
ENTERPASSWORD: akash
read buffer
substring before match
String
match + rest of string
LATITUDE
LONGITUDE
UTC-TIME
temperature: 0
tagstatus

Figure 7: Screenshot of the receiver system with password protection

2.4 Experimental setup

The **Error! Reference source not found.** shows the experimental setup of the portable data recorder. The device has to be set in an open place in order to obtain GPS coordinates and the main objective of the experiment is to properly encrypt the data with the secret key and to verify with the same secret key. To prove the data integrity and authenticity, the HMAC system was compared with the data recorder with no encryption and with a password based system.

Therefore the cases for the experiment were devised as follows:

1) No encryption performed and transmission to the receiver's GUI
2) No encryption perform and transmission of data to receiver's GUI which has password protection.
3) Encryption performed and sent to the receiver

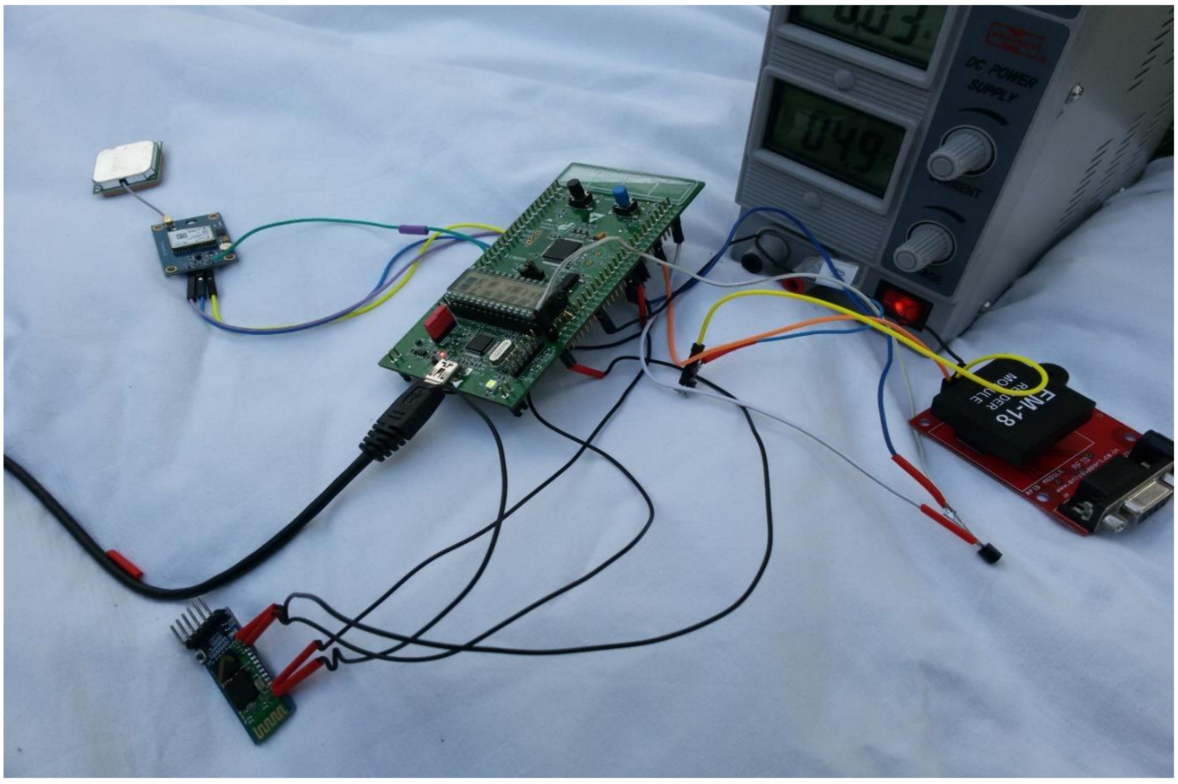


Figure 8: Prototype and experimental setup of the recorder

In the first case, the data was made to be obtained by the recorder and directly stored into the device memory. In this case, there was no encryption performed on the data. One of the reasons why this case was selected because, some of the personnel who handle the receiver system believes that, if they have the graphical user interface for their system, only they will be a valid person to view this data. In such a case, the personnel make a wrong assumption. The reason is because, the attack always pose threat when the data is unencrypted and transmitted. The attacker doesn't need a graphical user interface to view the output. With a simple terminal program software, the attacker can get the data that he needs. The above discussed case is given in Figure 9.

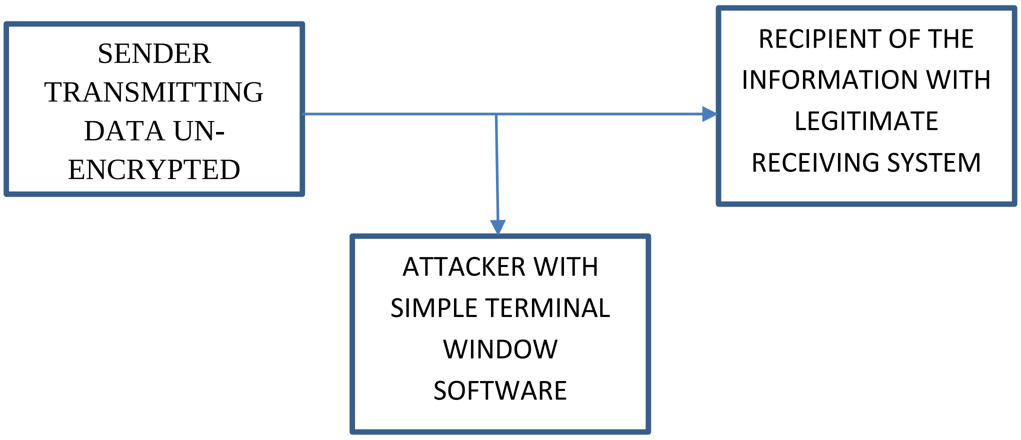

SENDER TRANSMITTING DATA UN-ENCRYPTED
RECIPIENT OF THE INFORMATION WITH LEGITIMATE RECEIVING SYSTEM
ATTACKER WITH SIMPLE TERMINAL WINDOW SOFTWARE

Figure 9: Transmitting unencrypted data to the recipient

The second case was for transmitting the data which is unencrypted to the recipient with the password protected graphical user interface. As per the research, there are certain companies which deal with confidential data and such companies do not have a terminal program software installed to any of the systems or pose strict norms to the employees on not bringing their own laptops to the workplace fearing from the employees with split personality. The personnel might believe a simple password will be enough to protect the data. There for to test the assumption of the personnel, the case was selected and implemented. The user interface at the receiver end was created in such a way that, only if the entered password is correct, the data can be viewed. But there is no powerful attack other than 'Brute force attack'. Since the data is transmitted through Bluetooth, the person sitting in the next system with user interface can try to crack the password and obtain the data from the recorder. This case is given in Figure *10*.

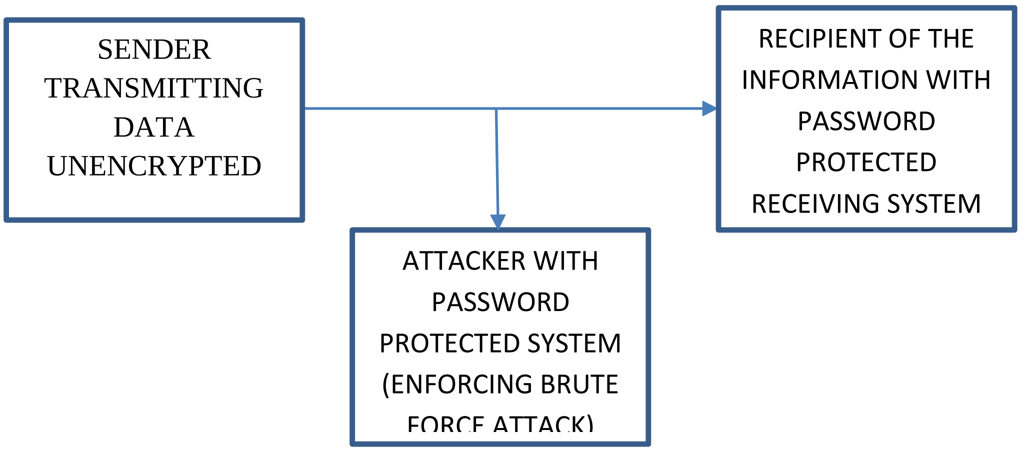

SENDER TRANSMITTING DATA UNENCRYPTED
RECIPIENT OF THE INFORMATION WITH PASSWORD PROTECTED RECEIVING SYSTEM
ATTACKER WITH PASSWORD PROTECTED SYSTEM (ENFORCING BRUTE FORCE ATTACK)

Figure 10: Data sent unencrypted to the password protected receiving system

In the third case, the data was signed using HMAC, where a secret key was used along with a hash function to deliver data integrity and authenticity. This secret key is a symmetric key, which mean, the key will only be shared between the sender and the receiver. The sender here in this case is the data recorder. The data recorder is programmed in such a way that the message is signed and a MAC is apprehended behind the message. This data is then sent to the receiver. The receiver, in order to verify uses his key to sign the message, compares the hash value. If both the hash values are equal, only then the message is accepted as valid. If not, the message is considered as invalid and then discarded. The attack with a legitimate user interface or even with a terminal window software cannot obtain the real text that is being transmitted. If the message is being attacked. After the message will be received, the receiver of the message gets to know that the message was attacked. The process is given in the Figure 11.

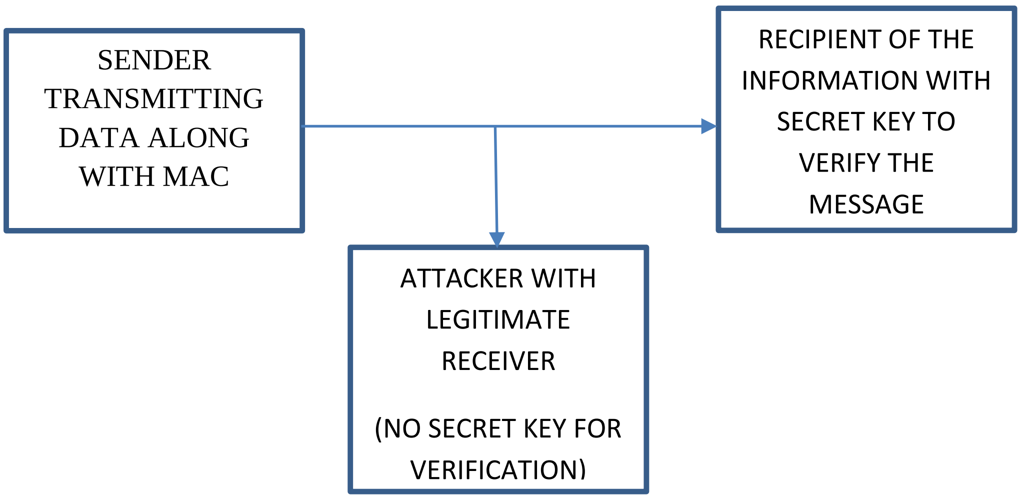
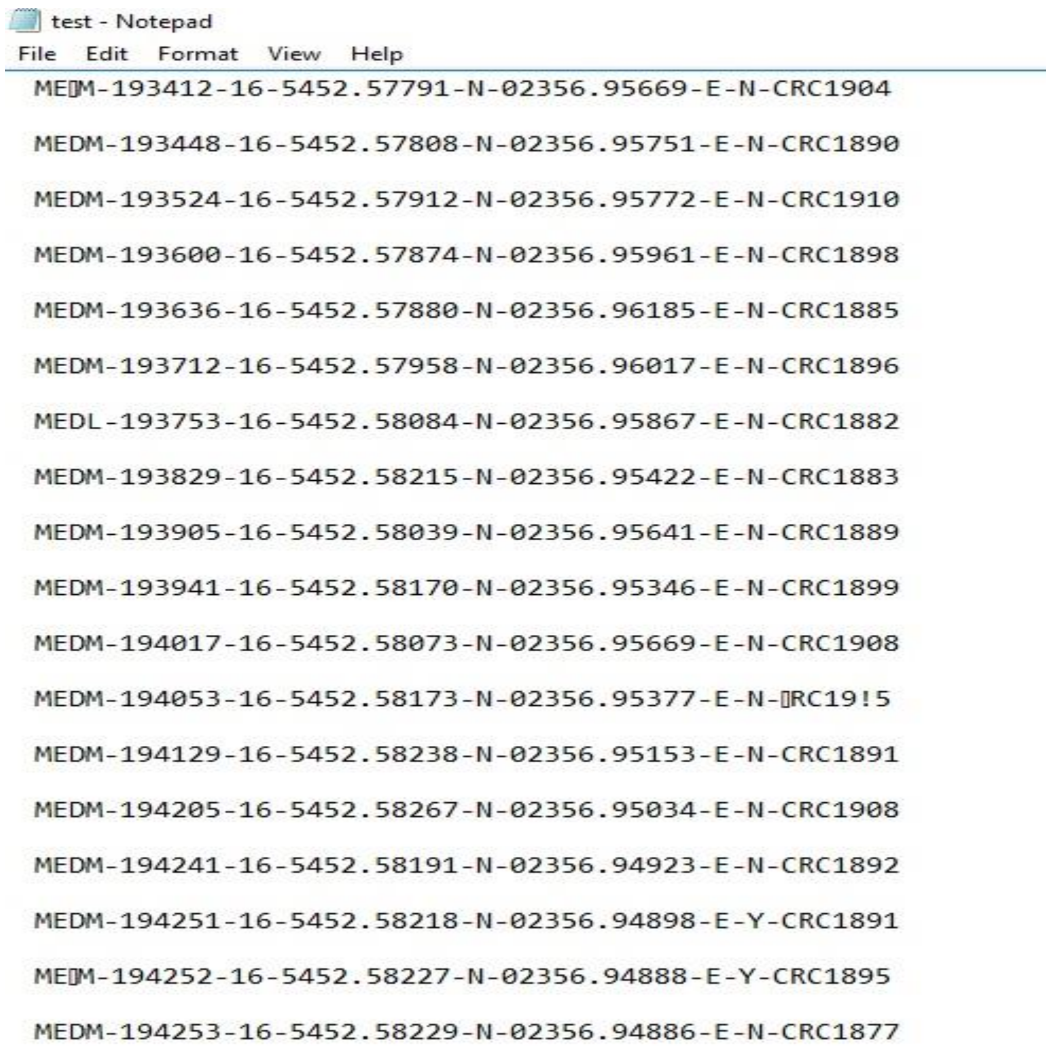

SENDER TRANSMITTING DATA ALONG WITH MAC
RECIPIENT OF THE INFORMATION WITH SECRET KEY TO VERIFY THE MESSAGE
ATTACKER WITH LEGITIMATE RECEIVER
(NO SECRET KEY FOR VERIFICATION)

Figure 11: Data sent along with MAC to the receiver

3. Results and observations

There were tests made in the beginning to know whether the data recorder was working properly and obtaining all the much needed data to its memory. The device was working successfully and these was the result for the below observation on the working of the data recorder.

1) The peripherals interfacing was perfect as all of them were obtaining data and were feeding it to the microcontroller's memory.
2) It was found that the GPS was the most important interface, as if GPS didn't receive coordinates, the output strings were not obtained for storing in the memory.
3) The output string was calculated to be 53 bytes output.



```
ME▯M-193412-16-5452.57791-N-02356.95669-E-N-CRC1904
MEDM-193448-16-5452.57808-N-02356.95751-E-N-CRC1890
MEDM-193524-16-5452.57912-N-02356.95772-E-N-CRC1910
MEDM-193600-16-5452.57874-N-02356.95961-E-N-CRC1898
MEDM-193636-16-5452.57880-N-02356.96185-E-N-CRC1885
MEDM-193712-16-5452.57958-N-02356.96017-E-N-CRC1896
MEDL-193753-16-5452.58084-N-02356.95867-E-N-CRC1882
MEDM-193829-16-5452.58215-N-02356.95422-E-N-CRC1883
MEDM-193905-16-5452.58039-N-02356.95641-E-N-CRC1889
MEDM-193941-16-5452.58170-N-02356.95346-E-N-CRC1899
MEDM-194017-16-5452.58073-N-02356.95669-E-N-CRC1908
MEDM-194053-16-5452.58173-N-02356.95377-E-N-▯RC19!5
MEDM-194129-16-5452.58238-N-02356.95153-E-N-CRC1891
MEDM-194205-16-5452.58267-N-02356.95034-E-N-CRC1908
MEDM-194241-16-5452.58191-N-02356.94923-E-N-CRC1892
MEDM-194251-16-5452.58218-N-02356.94898-E-Y-CRC1891
ME▯M-194252-16-5452.58227-N-02356.94888-E-Y-CRC1895
MEDM-194253-16-5452.58229-N-02356.94886-E-N-CRC1877
```

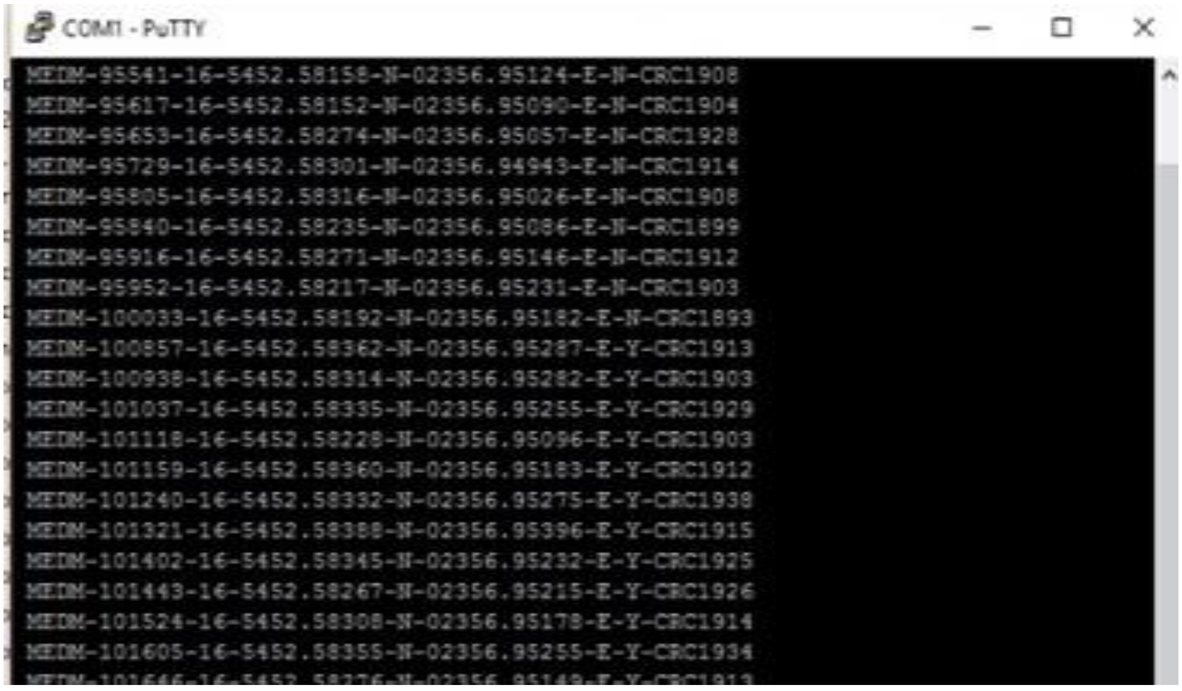
Figure 12: Data stored in a simple text file on the PC

The above Figure 12 shows the output of the data recorder stored in a text file on a PC. This was possible because, the recorder was programmed to have the option to select a file and path to store the incoming data. The first bits apprehended to the recorder is its preamble ‘MEDM’. Followed by the preamble, is the time and it was programmed to be obtained from GPS receiver itself. Following time is the temperature value. The bytes following temperature value is the GPS coordinates. The first one denotes latitude and the other one denotes longitude. Following GPS coordinates, the RFID tag status was read and written to the memory. Finally CRC was calculated for data integrity, to know whether the data is being properly sent and received.

As the recorder operation was considered to be successful, the next step was to analyse the results for the cases which were devised to compare the data authenticity and integrity.

CASE 1: TRANSMITTING UNENCRYPTED DATA

The recorder was made to send the data unencrypted to the receiver. One with a legitimate receiving system and other with a terminal window software. So as we know, the data was successfully received into the receiving system. The data were not only received by the receiving system, but the data was also received in the terminal window, which signs the end for data security and the compromising of integrity and authenticity of the data. The Figure 13 shows how the hacking was made possible.


COM1 - PuTTY

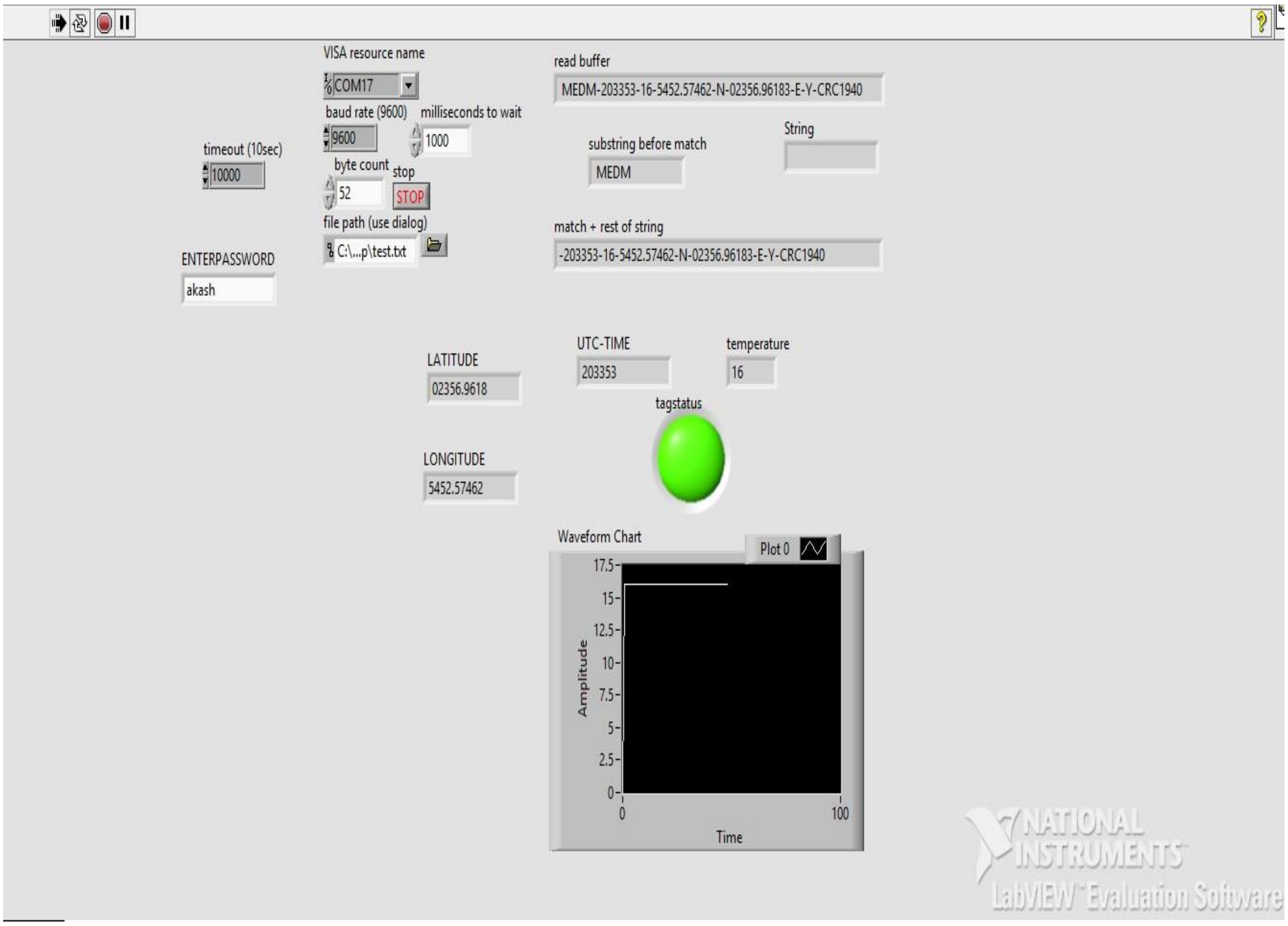
```
MEDM-95541-16-5452.58158-N-02356.95124-E-N-CRC1908
MEDM-95617-16-5452.58152-N-02356.95090-E-N-CRC1904
MEDM-95653-16-5452.58274-N-02356.95057-E-N-CRC1928
MEDM-95729-16-5452.58301-N-02356.94943-E-N-CRC1914
MEDM-95805-16-5452.58316-N-02356.95026-E-N-CRC1908
MEDM-95840-16-5452.58235-N-02356.95086-E-N-CRC1899
MEDM-95916-16-5452.58271-N-02356.95146-E-N-CRC1912
MEDM-95952-16-5452.58217-N-02356.95231-E-N-CRC1903
MEDM-100033-16-5452.58192-N-02356.95182-E-N-CRC1893
MEDM-100857-16-5452.58362-N-02356.95287-E-Y-CRC1913
MEDM-100938-16-5452.58314-N-02356.95282-E-Y-CRC1903
MEDM-101037-16-5452.58335-N-02356.95255-E-Y-CRC1929
MEDM-101118-16-5452.58228-N-02356.95096-E-Y-CRC1903
MEDM-101159-16-5452.58360-N-02356.95183-E-Y-CRC1912
MEDM-101240-16-5452.58332-N-02356.95275-E-Y-CRC1938
MEDM-101321-16-5452.58388-N-02356.95396-E-Y-CRC1915
MEDM-101402-16-5452.58345-N-02356.95232-E-Y-CRC1925
MEDM-101443-16-5452.58267-N-02356.95215-E-Y-CRC1926
MEDM-101524-16-5452.58308-N-02356.95178-E-Y-CRC1914
MEDM-101605-16-5452.58355-N-02356.95255-E-Y-CRC1934
MEDM-101646-16-5452.58276-N-02356.95149-E-Y-CRC1913
```

Figure 13: reception of data illegally using terminal window

In this case, there is no need for any secret key, massive algorithm to break or any type of attack needed. All that is needed is a terminal window software. Once the Bluetooth module of the recorder connects with the intended receiving system, it also automatically connects with the other Bluetooth modules, which is programmed as a slave. So without any effect or work put on attacking the system, the attacker is able to compromise the data which is being transmitted.

CASE 2: TRANSMITTING UNENCRYPTED DATA TO A PASSWORD PROTECTED SYSTEM

The Figure *14* shows the receiver system with password protection. The system is programmed in such a way that the data can be only viewed by the system is the entered password is correct. When a wrong password was entered into the system, there were no output shown. But in order to hack into the system like this, the brute force attack can get the work done, where all the possibilities can be tried.



VISA resource name: COM17
baud rate (9600): 9600
milliseconds to wait: 1000
timeout (10sec): 10000
byte count: 52
stop: STOP
file path (use dialog): C:\...p\test.txt
ENTERPASSWORD: akash
read buffer: MEDM-203353-16-5452.57462-N-02356.96183-E-Y-CRC1940
substring before match: MEDM
String:
match + rest of string: -203353-16-5452.57462-N-02356.96183-E-Y-CRC1940
LATITUDE: 02356.9618
LONGITUDE: 5452.57462
UTC-TIME: 203353
temperature: 16
tagstatus

Figure 14: Screenshot of the password protected Receiver system

If the attacker is someone who knows the personnel, who operates the system, then the possibility of attacking the system gets easier than usual. But if a password has numbers and symbols along with the alphabets, it tends to get hard for the attacker to crack. The attacker can use another type of brute for attack, which is called as dictionary attack, where the all the words in dictionary used can be tried out for a password. If even this doesn’t succeed, the attacker go with the special type of attack called as a key logger attack, where the hacker uses all the possibility to track all of the user’s key stroke [17]. With everything the user has typed-including his login ID’s and passwords have been recorded. This type of attack is different from brute force attack and dictionary attack. The key logger attack is used for stronger passwords because stronger passwords do not provide much protection against them. In this case, since a simple password was used, the brute force attack was enough to crack the password and to download the data to the attackers system.

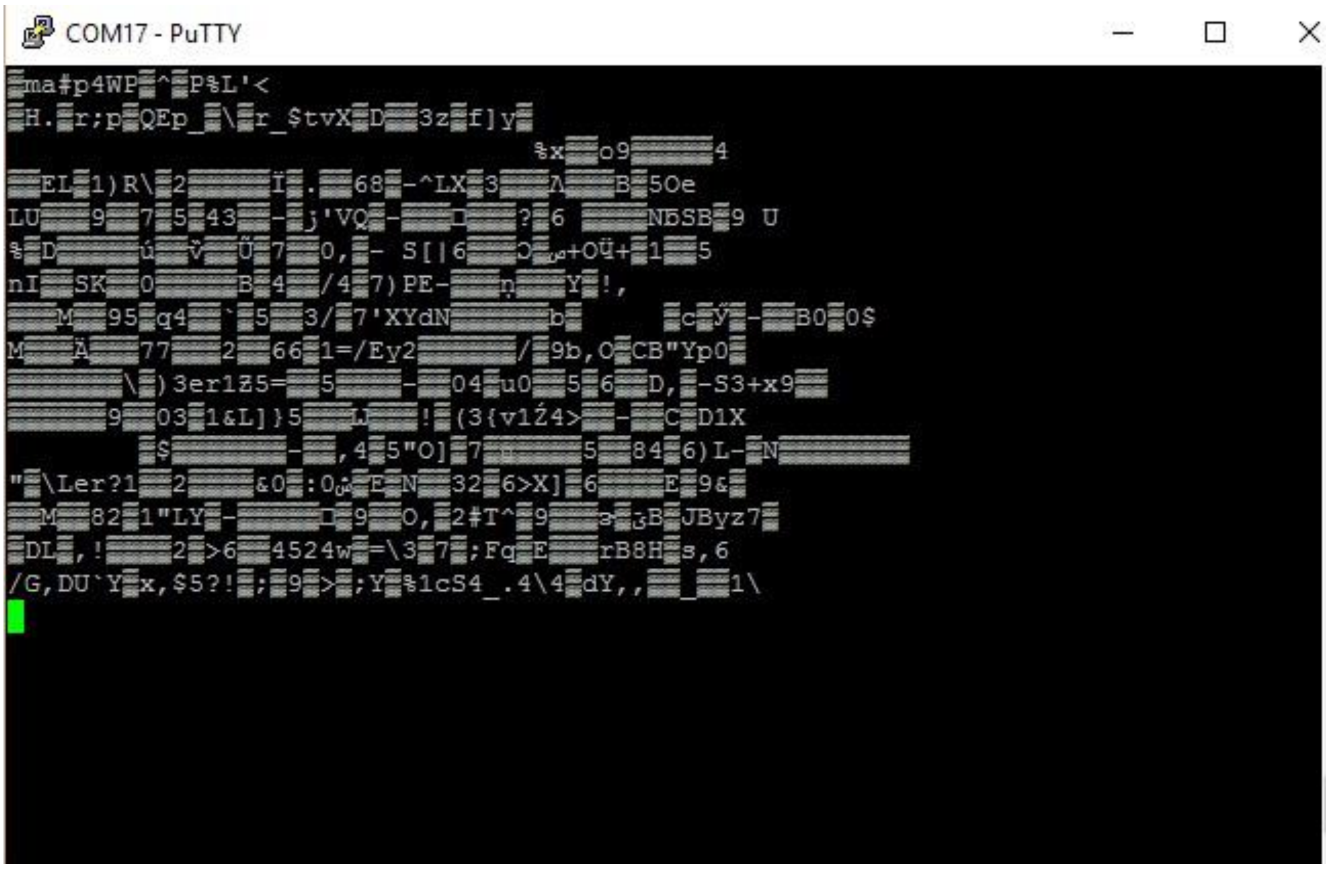
A point to note: even if a password system was used, there is no big need to go for any attacks as the data being received was unencrypted. Again, a simple terminal window software is enough to obtain the data to the attacker’s system. But, the stakes were raised in this case as the assumption was made for the companies which don’t use terminal window software in their computers.

There were only 3 to 4 attempts made by the attacker to obtain the data in this case.

CASE 3: SENDING DATA WITH A MAC AND USING A PROPER SYSTEM TO VERIFY THE MESSAGE WITH SHARED SECRET KEY.

In this case, the message is encrypted with a shared secret key and a hash function to compute Message Authentication Code (MAC) and this MAC is apprehended behind the message and transmitted to the receiver end. At the receiver end, the received message will be verified using the receiver’s secret key. This can be know, only if the secret key is known. The attacker in the middle will not be able to see the transmitted message clearly. The text will be only seen as the cipher text and without proper key, he cannot read the message.

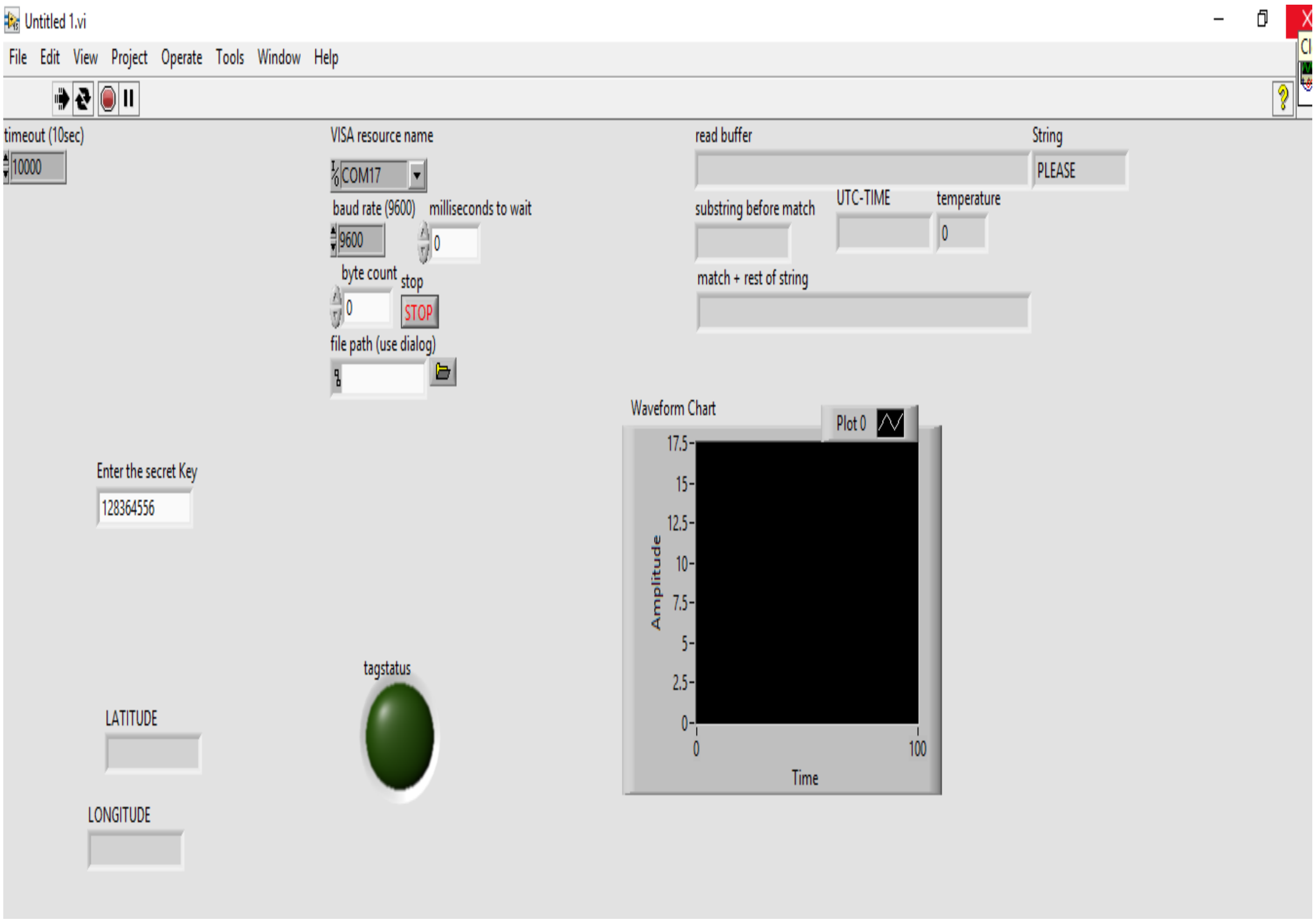
The given Figure 15 shows the output viewed in a terminal window, while trying to hack the message which is being sent. The viewed message is the cipher text of the real message along with the message authentication code. In order to crack this algorithm, the hacker needs to follow a complex task. Even if he manages to find out the secret key, with knowing what king of encryption method he is dealing with, it is difficult for him to decode the message. But with the secret key acquired, it might take time to find out the encryption technique, but it won’t be very long for the attacker to find the technique and decode the message. But the MAC attached to the message will provide extra security to the message.



```
▒ma‡p4WP▒^▒P%L'<
▒H.▒r;p▒QEp_▒\▒r_$tvX▒D▒▒3z▒f]y▒
                             %x▒▒o9▒▒▒▒▒4
▒▒EL▒1)R\▒2▒▒▒▒▒Ï▒.▒▒68▒-^LX▒3▒▒▒Λ▒▒▒B▒5Oe
LU▒▒▒9▒▒7▒5▒43▒▒-▒j'VQ▒-▒▒▒D▒▒▒?▒6 ▒▒▒▒NБSB▒9 U
%▒D▒▒▒▒▒ú▒▒ṽ▒▒Ű▒7▒▒0,▒- S[|6▒▒▒C▒ص+OÜ+▒1▒▒5
nI▒▒SK▒▒0▒▒▒▒▒B▒4▒▒/4▒7)PE-▒▒▒ṇ▒▒▒Y▒!,
▒▒▒M▒▒95▒q4▒▒`▒5▒▒3/▒7'XYdN▒▒▒▒▒▒b▒     ▒c▒Ў▒-▒▒B0▒0$
M▒▒▒Ä▒▒▒77▒▒▒2▒▒66▒1=/Ey2▒▒▒▒▒▒/▒9b,O▒CB"Yp0▒
▒▒▒▒▒▒\▒)3er1Ƶ5=▒▒5▒▒▒▒▒-▒▒04▒u0▒▒5▒6▒▒D,▒-S3+x9▒▒
▒▒▒▒▒▒9▒▒03▒1&L]}5▒▒▒Џ▒▒▒!▒(3{v1Ź4>▒▒-▒▒C▒D1X
       ▒$▒▒▒▒▒▒▒-▒▒,4▒5"O]▒7▒▒▒▒▒▒▒5▒▒84▒6)L-▒N▒▒▒▒▒▒▒▒
"▒\Ler?1▒▒2▒▒▒▒▒&0▒:0ش▒E▒N▒▒32▒6>X]▒6▒▒▒▒▒E▒9&▒
▒▒M▒▒82▒1"LY▒-▒▒▒▒▒▒D▒9▒▒O,▒2‡T^▒9▒▒▒▒з▒ق B▒JByz7▒
▒DL▒,!▒▒▒▒▒2▒>6▒▒4524w▒=\3▒7▒;Fq▒E▒▒▒▒rB8H▒s,6
/G,DU`Y▒x,$5?!▒;▒9▒>▒;Y▒%1cS4_.4\4▒dY,,▒▒▒_▒▒1\
```

Figure 15: Terminal window of the attacker

The Figure 16 shows the GUI of the attacker. Here, the entered secret key is incorrect. As the result the verification of the message cannot be achieved. The receiver system does not show any data because of improper verification. Here, the message obtained by the receiver will compute the hash function with the entered secret key. Where, in this case, the entered secret key s incorrect. As the result, the computed hash from the message along with the wrong secret key will not be equal to the hash ‘h’ obtained with the message received. Hence, this message will be considered invalid and the message will be discarded by the receiver.



timeout (10sec): 10000
VISA resource name: COM17
baud rate (9600): 9600
milliseconds to wait: 0
byte count: 0
String: PLEASE
temperature: 0
Enter the secret Key: 128364556

Figure 16: Receiver system of the attacker with wrong secret key entered

When viewed in the terminal window, the hash of the message does not look clear. Apparently, the hash of any message tends to appear as '4ed3ab12345f'. Therefore, a theoretical calculation was made on how the hash of the message should look for any message which is been sent by the portable data recorder.

For example, 5 messages are taken and hash for them are computed.

M1 = MEDM-193412-16-5452.57791-N-02356.95669-E-N-CRC1904

M2 = MEDM-193600-16-5452.57874-N-02356.95961-E-N-CRC1898

M3 = MEDM-193712-16-5452.57958-N-02356.96017-E-N-CRC1896

M4 = MEDM-194251-16-5452.58218-N-02356.94898-E-Y-CRC1891

M5 = MEDM-194252-16-5452.58227-N-02356.94888-E-Y-CRC1895

The secret key was selected as ‘12345’ and the method of generating HMAC was selected as SHA-1 algorithm.

Since the key is less than that of the message, the key was initially padded and the HMAC was generated using the software for the above messages.

The HMAC (in hex) for the messages were computed as follows:

HMAC1 = 7b0ecdd39313e788507af676a395814dcbd57f4c

HMAC2 = 3522239ded850ed9b26ad7ae64a681daa9ad4358

HMAC3 = 784eddfa0497f71e25075a38b906b3572d8e60d4

HMAC4 = 5c72916f19908c0b3d6087ad070ec84b39652f1e

HMAC5 = 12e14af02bdb924ea7409f05b3af77bbea53ab26

A comparison was made in order to show the outcomes of the experiment and to show without doubt, which case was providing the much needed integrity and authenticity to the data using Table 3.

The case 1 is shown as the unencrypted receiver system, case 2 is represented by password protected receiver and case 3 was represented with receiver system with HMAC. The comparison was made for use of any secret key, level of security, complexity of the attack, types of attack and to finally check whether the data integrity and authenticity was provided or not.

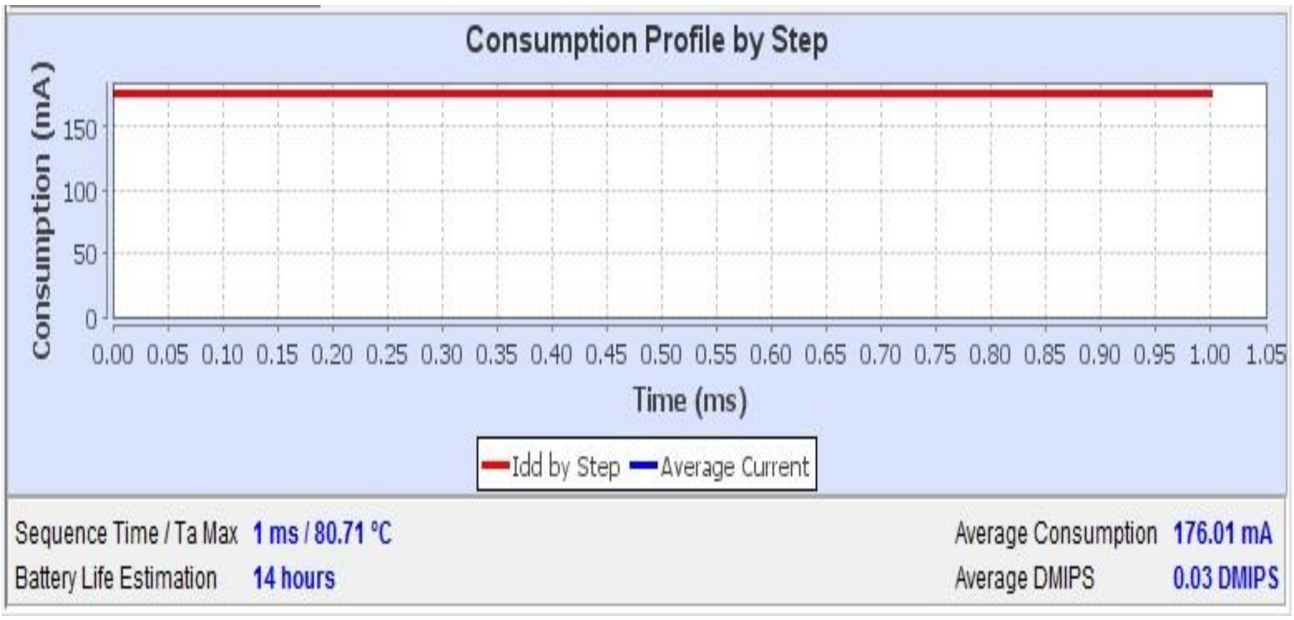
With the comparisons made, it has come to the point where the HMAC was providing much needed security to the messages sent than the password protected system. But, the combination of password protected system along with combination of HMAC can be a superior receiver as the attack has to be first made on the password and only then on the secret key, which now poses a double security to the transmitted data.

PARAMETERS	UNENCRYPTED RECEIVER SYSTEM	PASSWORD PROTECTED RECEIVER SYSTEM	RECIEVER SYSTEM WITH HMAC IMPLEMENTATI ON
USE OF SECRET KEY	No secret key was used	No secret key was used	Symmetric key was used
LEVEL OF SECURITY	Low level	Low level	High level
ATTACK COMPLEXITY	Easy to attack	Easy to attack, but some effort was needed	The message couldn’t be cracked easily
TYPES OF ATTACK	Use of simple terminal window software	Brute force attack, dictionary attack and key logger attack	Brute force attack, length extension attack and full key recovery attack
WAS DATA INTEGRITY AND AUTHENTICIT Y PROVIDE	It was not provided	It was not provide	Yes, it was provided to the message.

Table 3: Comparison of the devised cases

In addition to this, the power consumption of the device was estimated for time intervals of 1 min, 5 mins and 10 mins. In order to calculate power, the current consumption by the device was measured first. The current consumption was estimated between the board and for 2600 Ah Li Ion battery. The current consumption graph for the device is given below in the figure using STMicroelectronics Cube software. The current estimation during low power mode is 176.1mA.

1) Power consumption for every minute = 0.88 W/h
2) Average power consumption at every 5 minute interval = 176 mW/h
3) Average power consumption at every 10 (9 mins for better duty cycle) minutes interval = 88 mW/h



Sequence Time / Ta Max 1 ms / 80.71 °C
Battery Life Estimation 14 hours
Average Consumption 176.01 mA
Average DMIPS 0.03 DMIPS

Figure 17: Current consumption in low power mode

In the case of 5 minutes interval, the microcontroller sleeps for 5 minutes and then operates for every one minute after the minute interval and goes to sleep again. So the power is calculated based on the duty cycles, where the microcontroller works for 10 minutes on and 50 minutes off per hour. So the average power was estimated as 176 mW/h.

In case of 10 minutes interval, the microcontroller sleeps for 9 minutes and wake up for one minute to acquire the data and transmit them and again go back to sleep. Here, the controller operates for 6 minutes and sleeps for 54 minutes per hour. Therefore the average power was estimated as 88 mW/h.

In case of low power sleep mode, the estimated power during every 5 minutes and 10 minutes of sleep is given as below:

Since this is pulsed operation,

$$P = V*I$$

$$P = (5.49*10^{-6} * 5)$$

$$= 0.02745 \text{ mW}$$

1) Average power for every 5 minutes of sleep in an hour = 0.022 mW/h
2) Average power for every 10 minutes of sleep in an hour = 0.024 mW/h

A point to note, power consumption depends on the crystal oscillator speed. The battery life during real time measurement was 6 hrs and 45 mins.

4 Conclusion

- The security of the system relies on HMAC and key agreement protocol between recorder and receiver.
- Agreed symmetric keys should be securely stored in the recorder (in protected memory either password of pin protected memory).
- Key agreement protocol should be realised using diffie-helman 2048 moduli operation when p=2048 bits or with RSA key agreement protocol according to PKCS11 with |m| = 2048 bits.
- Secure HMAC functions can be as follows
  1) AES -128 CBC mode HMAC with shared key (K)
  2) Other standard HMAC functions.
- Realized transportation data consisting of time, GPS coordinates, temperature and position indicator.
- Position fixation allows to reduce the sample rate to
  1) Decrease sample rate and to reduce power consumption.
  2) Since adversary does not know, for what time interval the measurements are made.
- The experiment of the cases and their results are given below
  1) In the first case, it was observed that the data was easily compromised. There was no need for even a legitimate receiving system to obtain the data. The attacker was able to read the data only with a terminal program software.
  2) In the second case, it was observed the data was compromised, but was not compromised that easily. The brute for attack had to be employed to crack the password. The other attacks could have either been dictionary attack or key logger attack. Here, the attacker used a legitimate receiver system and cracked password with brute force.
  3) In the third case, the message was apprehended with MAC, which is technically hashing the message with a secret key. This message was not easy for the attacker to attack. The attacker had to go with complex algorithm or hacking technique in order to compromise the data.
- Here are the other following conclusions, which were made during the research of this project.

A) The power usage by the board was estimated to be 0.88 W/h

B) The charging time for the battery was estimated to be 5 hours

C) The battery lifetime estimated for the operation will be 14 hrs.

D) It was also observed that the GUI will only be operational, if the right password was entered, which in turn acts as the first line of defence for the incoming data's.

E) The entire device was powered up by a 5 v battery, through USB interface.

- There can be some of the updates made on the hardware to make it more effective and less power consuming like:

A) An NFC reader/writer can be used be used, if the data has to be written into some tag and SPBT2AT2 version Bluetooth module can be used instead of HC-05.

References

[1] “Fraud in Maritime industry,” [Online]. Available: http://www.skuld.com/documents/topics/cargo/fraud/fraud.pdf?epslanguage=en.

[2] “Data Encryption system,” [Online]. Available: https://en.wikipedia.org/wiki/Data_Encryption_Standard.

[3] “Advanced Encryption system,” [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard..

[4] “MD5 Algorithm,” [Online]. Available: https://en.wikipedia.org/wiki/MD5.

[5] “Message Digest 5,” [Online]. Available: http://docstore.mik.ua/orelly/other/Docs/oreilly/other2/puis3rd/0596003234_puis3-chp-7-sect-4.html.

[6] “Public key cryptosystems,” [Online]. Available: https://www.google.lt/search?q=public+key+and+private+key+cryptography&biw=1366&bih=643&source=lnms&tbm=isch&sa=X&sqi=2&ved=0ahUKEwjs_Y_bqvXMAhWKVSwKHRPCBHwQ_AUIBigB#imgrc=L9SEM_iojITeyM%3A.

[7] “Symmetric key Cryptography,” [Online]. Available: https://www.google.lt/search?q=private+key+cryptography&biw=1366&bih=643&source=lnms&tbm=isch&sa=X&sqi=2&ved=0ahUKEwj8ou2GqvXMAhUFjCwKHZgCAcsQ_AUIBigB#imgrc=WFt-WK2uDdCHfM%3A..

[8] M. Blumenthal, “Encryption: Strengths and Weaknesses of Public-key Cryptography”.

[9] “Threats to symmetric key encryption system,” [Online]. Available: https://www.cs.clemson.edu/course/cpsc424/material/Cryptography/Attacks%20on%20Symmetric%20Key.pdf..

[10] MGcGrew, Authenticated Encryption with AES-CBC-HMAC, 2009.

[11] R. R. module. [Online]. Available: http://www.nskelectronics.com/em-18_rfid_reader.html.

[12] G. Module. [Online]. Available: https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf.

[13] M. Ranjeet, S. Vivek, A. Jibi and M. Rajni, “ANALYSIS AND COMPARISON OF SYMMETRIC KEY CRYPTOGRAPHIC ALGORITHMS BASED ON VARIOUS FILE FEATURES”.

[14] H.Krawczyk, M. Bellare and R. Canetti, “HMAC : Keyed-Hashing for Message Authentication Code, Network Working group,” [Online]. Available: https://tools.ietf.org/html/rfc2104..

[15] “Hash- based Message Authentication Code,” [Online]. Available: https://en.wikipedia.org/wiki/Hash-based_message_authentication_code.

[16] “Stop using unsafe keyed hashes, use HMAC,” [Online]. Available: https://rdist.root.org/2009/10/29/stop-using-unsafe-keyed-hashes-use-hmac/.

[17] T. Rizzo, “Insights in IT security,” [Online]. Available: http://insights.scorpionsoft.com/3-types-of-password-security-attacks-and-how-to-avoid-them.

Appendix

PROGRAM CODE

MCU PROGRAM

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "stm32l1xx_hal.h"

char  *TIMEZONE="053000";

static char *tagref="44001D22F883";

/* Private variables ---------------------------------------------------------*/

ADC_HandleTypeDef hadc;

RTC_HandleTypeDef hrtc;

UART_HandleTypeDef huart1;

UART_HandleTypeDef huart2;

UART_HandleTypeDef huart3;


/* USER CODE BEGIN PV */

/* Private variables ---------------------------------------------------------*/


/* USER CODE END PV */


/* Private function prototypes -----------------------------------------------*/

void SystemClock_Config(void);

static void MX_GPIO_Init(void);

static void MX_USART1_UART_Init(void);

static void MX_ADC_Init(void);

static void MX_USART2_UART_Init(void);
static void MX_USART3_UART_Init(void);

static void MX_RTC_Init(void);

static uint32_t convert_time(char*,char*);

static uint32_t crcval(char*);

static int compare(char*,char*);


/* USER CODE BEGIN PFP */

/* Private function prototypes -----------------------------------------------*/


/* USER CODE END PFP */


/* USER CODE BEGIN 0 */


/* USER CODE END 0 */


int main(void)

{


  /* USER CODE BEGIN 1 */


  /* USER CODE END 1 */


  /* MCU Configuration----------------------------------------------------------*/


  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */

  HAL_Init();
  /* Configure the system clock */

  SystemClock_Config();


  /* Initialize all configured peripherals */




  /* USER CODE BEGIN 2 */


  /* USER CODE END  2 */

  MX_RTC_Init();

  /* Infinite loop */

  /* USER CODE BEGIN WHILE */

  while (1)

  {


  }

  /* USER CODE END 3 */


}


uint32_t crcval(char *str)

{  uint32_t result=0;

        uint8_t var=1,character;

        while(*str!='\0')

        { character=*str;

                result=result+(character^var);

                str++;
		var++;
	}

	return result;
}

int compare(char *x,char*y)
{
	uint8_t temp=0;
	char c[1];
//HAL_UART_Transmit(&huart2,"inside\n\r",8,HAL_MAX_DELAY);
	while(*y!='\0')
	{
		if(*x!=*y)
		{
			temp++;
		}
		x++;
		y++;
	}
	sprintf(c,"%d",temp);
//	HAL_UART_Transmit(&huart2,c,1,HAL_MAX_DELAY);
//	HAL_UART_Transmit(&huart2,"complete\r\n",10,HAL_MAX_DELAY);
	return temp;
}

/** System Clock Configuration
```

```
*/

void RTC_Alarm_IRQHandler(void)

{

  /* USER CODE BEGIN RTC_Alarm_IRQn 0 */



  /* USER CODE END RTC_Alarm_IRQn 0 */

  HAL_RTC_AlarmIRQHandler(&hrtc);

              MX_GPIO_Init();

  MX_USART1_UART_Init();

  MX_ADC_Init();

  MX_USART2_UART_Init();

  MX_USART3_UART_Init();

   uint8_t adcval,temperature,tagstatus;

   uint8_t buffer[1];

            char tag[12];

            uint32_t tmp,count,utctime;

  char *temp,*recvstr,*rmcstr,*time,*latitude,*y,*longitude,*nors,*eorw;

  unsigned char packet[100],packetcrc[150];

            recvstr=malloc(500);

            HAL_UART_Transmit(&huart2,"RTC\r\n",5,HAL_MAX_DELAY);




                        memset(recvstr,0,500);

                        memset(packet,0,100);

                        memset(packetcrc,0,100);

                        memset(tag,0,12);

                        temp=recvstr;
		HAL_UART_Receive(&huart3,tag,12,5000);
//		HAL_UART_Transmit(&huart2,tag,12,HAL_MAX_DELAY);
		do
		{


			HAL_UART_Receive(&huart1,buffer,1,HAL_MAX_DELAY);
			*recvstr=buffer[0];
			recvstr++;
		}while(buffer[0]!='\n');
		recvstr=temp;
		rmcstr=strstr(recvstr,"GPRMC");
		y=strtok(rmcstr,",");
			time=strtok(NULL,",");
			y=strtok(NULL,",");
			latitude=strtok(NULL,",");
			nors=strtok(NULL,",");
		longitude=strtok(NULL,",");
		eorw=strtok(NULL,",");
		count=100;
		tmp=0;
		while(count<100)
		{
		HAL_ADC_Start(&hadc);
		adcval=HAL_ADC_GetValue(&hadc);
		HAL_ADC_Stop(&hadc);
		HAL_Delay(100);
			tmp=adcval+tmp;
```

```
count--;
}
temperature=((tmp/100)-240);



if(time!=NULL)
{
        if(latitude!=NULL)
        {
                if(longitude!=NULL)
                {
                        if(compare(tag,tagref)==0)
                        {
                        //HAL_UART_Transmit(&huart2,"yes\n\r",5,HAL_MAX_DELAY);
                                tagstatus='Y';

                        }
                        else
                        {
                        //HAL_UART_Transmit(&huart2,"No\n\r",5,HAL_MAX_DELAY);
                                tagstatus='N';
                        }
                        utctime=atoi(time);
                        sprintf(packet,"\r\n                    MEDM-%d-%d-%s-%s-%s-%s-
%c",utctime,temperature,latitude,nors,longitude,eorw,tagstatus);
                        sprintf(packetcrc,"%s-CRC%d",packet,crcval(packet));
```

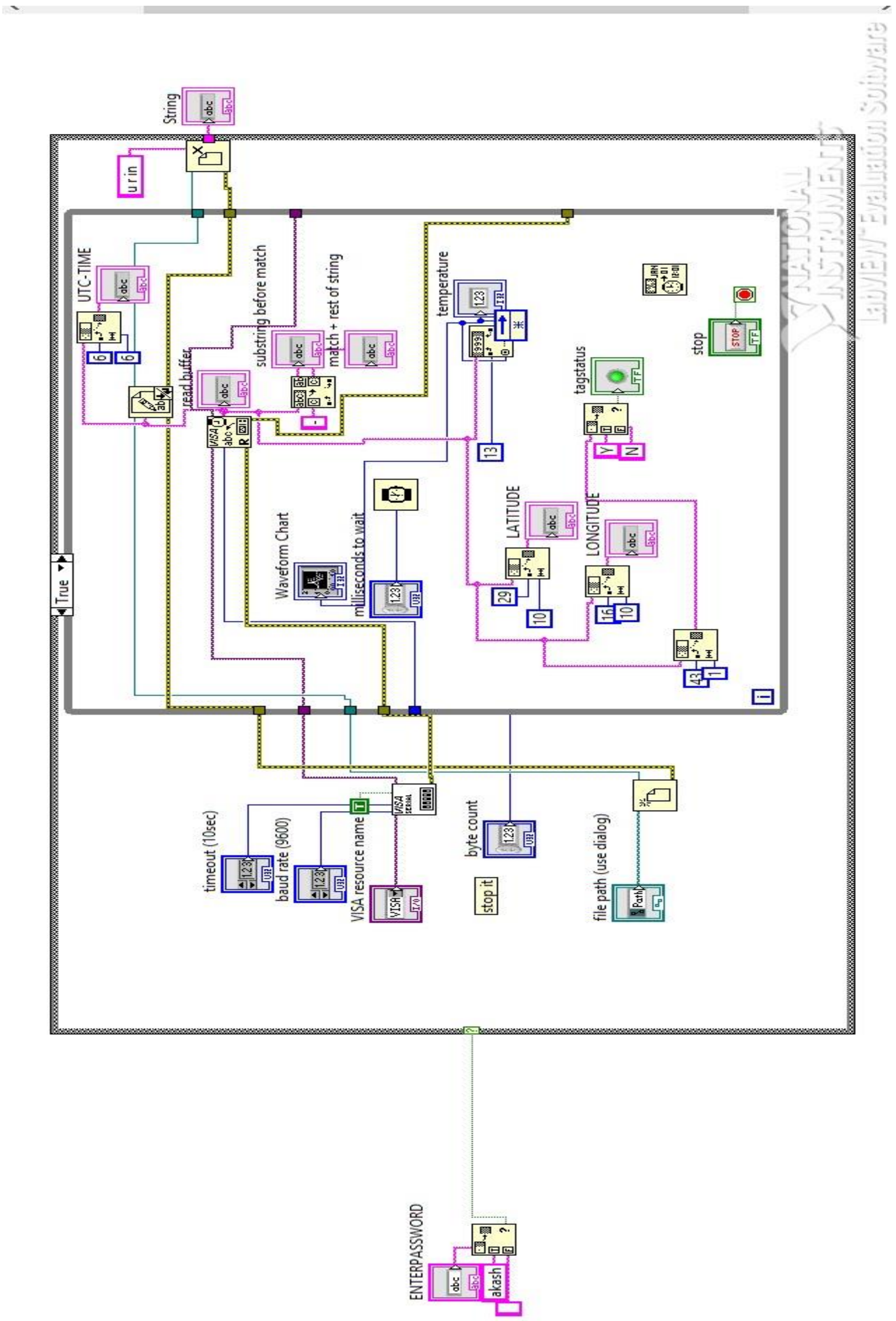
```
            HAL_UART_Transmit(&huart2,packetcrc,strlen(packetcrc),HAL_MAX_DELAY);
                                    }
                            }
                    }
            HAL_UART_DeInit(&huart1);
            HAL_UART_DeInit(&huart2);
            HAL_UART_DeInit(&huart3);
            /* USER CODE BEGIN RTC_Alarm_IRQn 1 */
  /* USER CODE END RTC_Alarm_IRQn 1 */
}
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInit;

  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_LSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL6;

  RCC_OscInitStruct.PLL.PLLDIV = RCC_PLL_DIV3;

  HAL_RCC_OscConfig(&RCC_OscInitStruct);


  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;

  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV128;

  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0);


  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;

  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;

  HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit);


  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);


  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);


  /* SysTick_IRQn interrupt configuration */

  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);

}


/* ADC init function */

void MX_ADC_Init(void)

{
```

```
ADC_ChannelConfTypeDef sConfig;

  /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)
  */
hadc.Instance = ADC1;
hadc.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
hadc.Init.Resolution = ADC_RESOLUTION_12B;
hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc.Init.ScanConvMode = ADC_SCAN_DISABLE;
hadc.Init.EOCSelection = ADC_EOC_SEQ_CONV;
hadc.Init.LowPowerAutoWait = ADC_AUTOWAIT_DISABLE;
hadc.Init.LowPowerAutoPowerOff = ADC_AUTOPOWEROFF_DISABLE;
hadc.Init.ChannelsBank = ADC_CHANNELS_BANK_A;
hadc.Init.ContinuousConvMode = DISABLE;
hadc.Init.NbrOfConversion = 1;
hadc.Init.DiscontinuousConvMode = DISABLE;
hadc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc.Init.DMAContinuousRequests = DISABLE;
HAL_ADC_Init(&hadc);

  /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
  */
sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = 1;
sConfig.SamplingTime = ADC_SAMPLETIME_4CYCLES;
```

```
  HAL_ADC_ConfigChannel(&hadc, &sConfig);

}

/* RTC init function */
void MX_RTC_Init(void)
{

  RTC_TimeTypeDef sTime;
  RTC_DateTypeDef sDate;
  RTC_AlarmTypeDef sAlarm;

    /**Initialize RTC and set the Time and Date
    */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  HAL_RTC_Init(&hrtc);

  sTime.Hours = 0x0;
  sTime.Minutes = 0x0;
  sTime.Seconds = 0x0;
  sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
  sTime.StoreOperation = RTC_STOREOPERATION_RESET;

  HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD);


  sDate.WeekDay = RTC_WEEKDAY_MONDAY;

  sDate.Month = RTC_MONTH_JANUARY;

  sDate.Date = 0x1;

  sDate.Year = 0x0;


  HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD);


    /**Enable the Alarm A

    */

  sAlarm.AlarmTime.Hours = 0x0;

  sAlarm.AlarmTime.Minutes = 0x0;

  sAlarm.AlarmTime.Seconds = 0x0;

  sAlarm.AlarmTime.SubSeconds = 0x0;

  sAlarm.AlarmTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;

  sAlarm.AlarmTime.StoreOperation = RTC_STOREOPERATION_RESET;

  sAlarm.AlarmMask = RTC_ALARMMASK_ALL;

  sAlarm.AlarmSubSecondMask = RTC_ALARMSUBSECONDMASK_ALL;

  sAlarm.AlarmDateWeekDaySel = RTC_ALARMDATEWEEKDAYSEL_DATE;

  sAlarm.AlarmDateWeekDay = 1;

  sAlarm.Alarm = RTC_ALARM_A;

  HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BCD);


}
```

```
/* USART1 init function */
void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_8;
  HAL_UART_Init(&huart1);

}

/* USART2 init function */
void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_8;
  HAL_UART_Init(&huart2);


}


/* USART3 init function */
void MX_USART3_UART_Init(void)
{


  huart3.Instance = USART3;
  huart3.Init.BaudRate = 9600;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  HAL_UART_Init(&huart3);


}
#ifdef USE_FULL_ASSERT


/**
   * @brief Reports the name of the source file and the source line number
   * where the assert_param error has occurred.
   * @param file: pointer to the source file name
   * @param line: assert_param error line source number
   * @retval None
   */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */

}

#endif

/**
  * @}
  */

/**
  * @}
*/
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

GUI CODE


String
u r in
UTC-TIME
6
6
read buffer
substring before match
match + rest of string
-
temperature
13
tagstatus
Y
N
stop
Waveform Chart
milliseconds to wait
LATITUDE
29
10
LONGITUDE
16
10
43
1
True
timeout (10sec)
baud rate (9600)
VISA resource name
byte count
stop it
file path (use dialog)
ENTERPASSWORD
akash
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software