View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CiteSeerX

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET)

ISSN 0976 – 6367(Print)
ISSN 0976 – 6375(Online)
Volume 4, Issue 1, January- February (2013), pp. 152-170
© IAEME: www.iaeme.com/ijcet.asp
Journal Impact Factor (2012): 3.9580 (Calculated by GISI)
www.jifactor.com

IJCET
©IAEME

SECURING TESLA BROADCAST PROTOCOL WITH DIFFIE-HELLMAN KEY EXCHANGE

Krishnakumar S1, Srinivasan R2
1Research Scholar, Dept of Computer Science & Engg, SRM University
Chennai (India), email: sendtokrishna@yahoo.co.in
2Professor, Dept of Computer Science & Engg, SRM University,
Chennai (India), email:rsv38@yahoo.co.in

ABSTRACT

Broadcast communication is highly prone to attacks from unauthenticated users in the wireless medium. Techniques have been proposed to make the communication more secure. In this paper, TESLA broadcast protocol is used to ensure source authentication. Diffie-Hellman Key Exchange is used to share the cryptographic keys in a secured manner. A PKI is developed based on TESLA and Diffie-Hellman Key Exchange, assuming that all network nodes in the network are loosely synchronized in time.

Keywords: Timed Efficient Stream Loss-tolerant Authentication (TESLA), Message Authentication Code (MAC), Diffie-Hellman Key Exchange, Denial of Service (DoS), Public Key Infrastructure

I. INTRODUCTION

The broadcast communication involves large scale spreading of data throughout the network. Some of the examples of them are Satellite broadcast, IP multicast, Wireless radio broadcast. There may be many unauthenticated users in the wireless network. To avoid them, the receiver has to ensure that the message it is receiving is from the original sender. An unknown user takes the identity of sender and injects broadcast packets. This phenomenon is known as packet injection attack. From the receiver's point of view, it doesn't know whether the message received was from an authenticated sender and was not altered en route. From the sender's point of view, it does not retransmit the lost packets, because of mutually untrusted receivers and unreliable communication environments.

TESLA authorizes all receivers to verify the integrity and authenticate the packet source in broadcast or multicast streams [21]. TESLA can be used in the transport layer, in the network layer, or in the application layer. TESLA generates different keys using the one-way key chain, but they need to be exchanged in a secured manner, which is done through Diffie-Hellman Key Exchange.

The Diffie-Hellman Key Exchange is a secure and robust method of exchanging the cryptographic keys. It involves public keys and secret keys which are exchanged between the sender and receiver. At the beginning of the process both the users does not know whether they have similar keys, it is only at the end they get the similar and thereby establishing secured transfer of the keys.

II. RELATED WORK

Some solutions have been proposed to increase the source authentication; but they do not satisfy the full requirements. The Point-to-Point Authentication mechanism is a straightforward technique. It involves attaching a MAC (Message Authentication Code), computed using a secret shared key, to each packet. This does not fulfill the needs, because any user with that shared key can take the form of the sender. An asymmetric cryptography can prohibit this attack; such an attempt is given in the next method.

In Digital signature scheme signed data packets are used. But, some of the demerits of this technique is, it has high overhead, in terms of time and verification of bandwidth. Also, it is computationally expensive. Denial-of-service attacks are a phenomenon where the sender is flooded with time synchronization requests. Request implosion is a problem where the sender is devastated with time synchronization requests from receivers.

To avoid these high overhead many schemes are suggested [10], [18], [20], [27], [28], but they crashed in:

- Bandwidth overhead
- Processing time
- Scalability
- Healthiness to denial-of-service attacks

A solution proposed by Canetti, *et al.,* [5] involves ‘k’ different keys and ‘k’ different MAC’s for every message. If every receiver, has ‘m’ keys, it can verify ‘m’ MAC’s. The keys are sorted so that there is no scission of ‘w’ receivers and it can advance a packet for a particular receiver. Security of this technique depends on the assumption that at most a qualified number (of the order ‘k’) of receivers conspire. A solution by Boneh, *et al.*, [4] suggest that it is impossible to build a compact connivance resistant broadcast authentication protocol, without neither depending on digital signatures nor on time synchronization. They have indicated that any assured broadcast authentication protocol, with per-packet overhead, bit less than number of receivers can be transformed into a signature scheme.

Also a Symmetric cryptography on MAC [7] depends on deferred disclosure of keys by the sender. This technique was discovered by Cheung in the space of affirming updates of link state routing. None of the techniques provided, could fully eliminate the problems in the broadcast communication, so the TESLA (Timed Efficient Stream Loss-tolerant Authentication) broadcast protocol is chosen. Some of the features of the TESLA protocol are:

- Low overhead in terms of computation and communication
- Low authentication delay, of the order of one round trip delay between the receiver and the sender
- Applicable for large number of receivers
- Good tolerance of packet loss
- Asymmetric cryptographic functions
- Bounded buffering required for the sender and the receiver

a. Main Idea of TESLA Protocol

Time is used for the asymmetry and also used as the key. The message is divided in 'n' packets and a MAC is appended to each packet. The MAC is computed through a key 'k', cognized only to the sender. Receiver cannot authenticate the message, so it has to buffer it for a while. When the sender opens up the key to the receiver, the packets are affirmed. The one condition is that the receiver has to synchronize its clock with the sender well ahead of time.

The needs of TESLA are:

- For the receivers to be synchronized loosely in time, a protocol is needed to achieve it.
- A quality mechanism to authenticate keys at the receiver end.

The schematic of the outline of the TESLA protocol is given in Fig. 1,

b. One-Way Chain

There is a need to commit to an order of random values, so a one-way chain is used. One-way hash function produces a one-way chain. The applications of one-way chains are one-time passwords [15] and S/KEY one-time password system [13].

The parameters to be used in the one-way chain construction are:

- 'l' – Length of the chain
- 's_l' – Last element of the chain
- F – One-way function
- s_0 – Commitment to the whole one-way chain (through which any element of the chain can be verified)

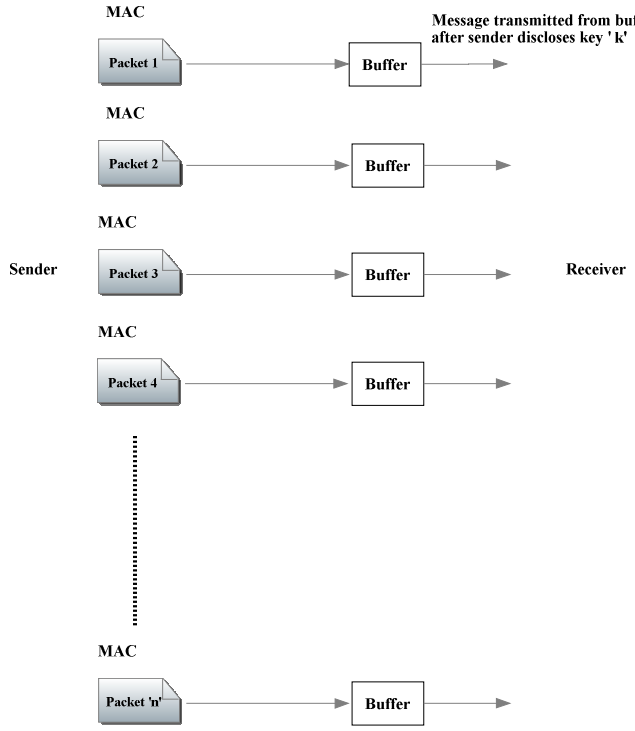


Fig. 1. Schematic Outline of TESLA protocol.

The construction of the one-way chain follows the following steps:

- Apply 'F' repeatedly to generate the chain
- From s_0, the element s_i of index 'i', can be checked. This is indeed the element with the index 'i' of the hash function. It is checked by verifying $F^i(s_i) = s_0$.
- Similarly, when i<j, s_i executes to s_j, by checking $F^{j-i}(s_j) = s_i$
- Thus, the entities in the chain are arranged in the ascending order $s_0, s_1, \ldots, s_{l-1}, s_l$

The storage of the one-way chain can be done in two ways. It can be generated all at once and stored. Else the last entity alone is kept and any other entity is calculated on request. But, in practical use, a hybrid approach decreases the storage with less recalculation penalty. One-way chain has an advantage that even if the middle values in the chain are lost, it can be recalculated using incoming values. So, even if some revealed keys are missing, a receiver can redeem the key chain and verify the correctness of the packets. The correlation of one-way Chain with TESLA is that the components of the one-way chain are keys, so it is known as one-way key chain. Also, any key of the one-way key chain can find out all the following keys.

The requirements of TESLA with respect to time are that, it should be loosely synchronized in time and the receiver must know an upper limit on the sender's local time. So, we go for time synchronization.

c. Time synchronization

Assume that the clock drift of the sender and the receiver is negligible; else the receiver can resynchronize the time with the sender, at regular intervals. The parameters used in the time synchronization process are:

- δ – difference in time between the sender and receiver
- Δ – The upper limit on δ, also known as *maximum time synchronization error*

Each receiver does definitive time synchronization with the sender. The advantage of this is it does not require any extra infrastructure for time synchronization. A two-round time synchronization protocol [22], [23], fulfills the need for TESLA, where the receiver knows an upper bound on sender's clock. The timing diagram of the time synchronization is given in Fig. 2,

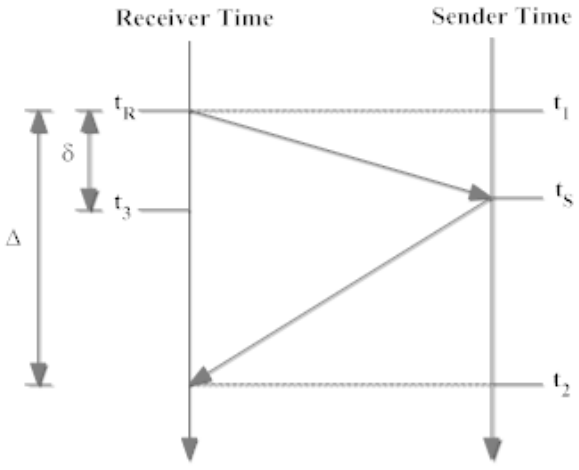


Fig. 2. Time Synchronization.

The receiver gives a time synchronization request at time t_R, when the sender's clock is at time t_1. Now, the sender replies to the request at its local time t_S. During the current time of receiver (t_r), the upper limit on the current sender's time is calculated as $t_s \leq t_r - t_R + t_S$. But, the receiver doesn't know about the propagation delay of the time synchronization request packet, so it is taken that time synchronization error is Δ (or the completer round-trip time (RTT)).

The receiver records its local time t_R and sends a time synchronization requesting containing a nonce to the sender. The delay of the processing and the propagation does not alter δ (under the assumption that the sender records and replies immediately with the arrival time of request packet).

In the setup process the sender S has a digital signature key pair, and private key K_S^{-1} and public key K_S. It is assumed that a mechanism allows a receiver R to learn the authenticated public key K_S. Then, the receiver takes an arbitrary and unpredictable nonce.

The steps of the protocol are:

- The receiver takes up its local time t_R.
- The receiver validates the digital signature and checks that nonce in packet equals to that of arbitrarily generated.
- If the message is original, the receiver stores t_R and t_S.
- To calculate the upper limit on the sender's clock at local time t, the receiver calculates $t - t_R + t_S$

On obtaining the signed response, receiver sees the validity of the signature and checks that the nonce in the response packet equals the nonce in the requested packet. If all the verifications are true, receiver uses t_R and t_S to calculate the upper limit on the sender's time.

d. Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange is a technique for exchanging cryptographic keys in a secured manner [8]. The interesting fact of this method is that the two users actually never get to choose the key, but at the end they would have calculated the same key, which is not easy for a hostile user to calculate. The Diffie-Hellman Key Exchange is based on the discrete (or) exponentiation problem. Given a base 'x', an exponent 'y' and a modulus 'z', calculate 'a' such that $x^y \equiv a \pmod z$, where $0 \leq a < z$. It seems that the problem is simple enough to solve and obtain 'a'. But when it comes to the inverse it is difficult to solve, i.e. given a base 'x', a result 'a', where $0 \leq a < z$ and a modulus 'z', calculate the exponent 'z' such that $x^y \equiv a \pmod z$. One can try to solve the problem by trying different values for the variables, but it is time consuming and tedious process especially for large prime number values of 'c'.

In Diffie-Hellman Key Exchange, two users "X" and "Y" agree on two values, a large prime 'i' and a generator 'j', where $1 < j < i$. These are the public values. But in secret, X selects a secret key 'a', with $1 < a < i$ and Y selects a secret key 'b', with $1 < b < i$. X calculates $j^a \pmod i$ and sends it to Y. This is known as f (a). Y calculates $j^b \pmod i$ and sends it to X. This is known as f (b). f (a) and f (b) are also public values. But, in secret X calculates f (b) ^{a} and Y calculates f (a) ^{b}, known as the exchanged keys. Both the exchanged keys values are same which is $j^{ab} \pmod i$, thus establishing secure exchange of the keys with any interruption from any hostile user.

III. THE TESLA PROTOCOL

This section explains the working of the TESLA protocol in detail.

e. Overview of TESLA Protocol

The receivers need to check the authentication information, but not to produce it. The sender divides the time into uniform intervals. Then, the sender builds a one-way chain of self-authenticating values, and allocates the values sequentially to the time periods, i.e. only one key per time period. The one-way chain is applied in the reverse order of generation, so that any value of a time period can be taken to derive values of previous time periods. The sender establishes a disclosure time for the one-way chain, usually on the order of few time periods. The sender declares the value after the disclosure time.

The sender appends a MAC to every packet. The MAC is calculated over the matter of the packet. For every packet, the sender evaluates the time period and applies the corresponding entity from the one-way chain as a cryptographic key to calculate the MAC. Additionally, the sender also dispatches the most recent one-way chain value that it can reveal. When every receiver receives the packet, it undergoes the following steps. It gains knowledge about the order of the revealing keys. Because the clocks are synchronized loosely in time, the receiver can see whether the key used to calculate the MAC is still not disclosed. It is checked by finding whether the sender could not yet have reached the time interval for revealing it. If the MAC key is still not disclosed, then the packet is being buffered by the receiver.

Every receiver checks if the revealed key is correct (utilizing self-authentication and keys released previously) and then checks the trueness of the MAC of packets buffered, that were sent in the time period of the revealed key. If the MAC is genuine, the receiver takes the packet. On notation terms, the stream of messages to be distributed by the sender is denoted by $\{M_i\}$, the network packet along with the authentication details is denoted by P_i. The broadcasting channel may be lossy, but the sender does not propagate the packets again. Even though there is some packet loss, every receiver needs to approve all the messages it receives.

The working of the TESLA protocol is described in Fig. 3,

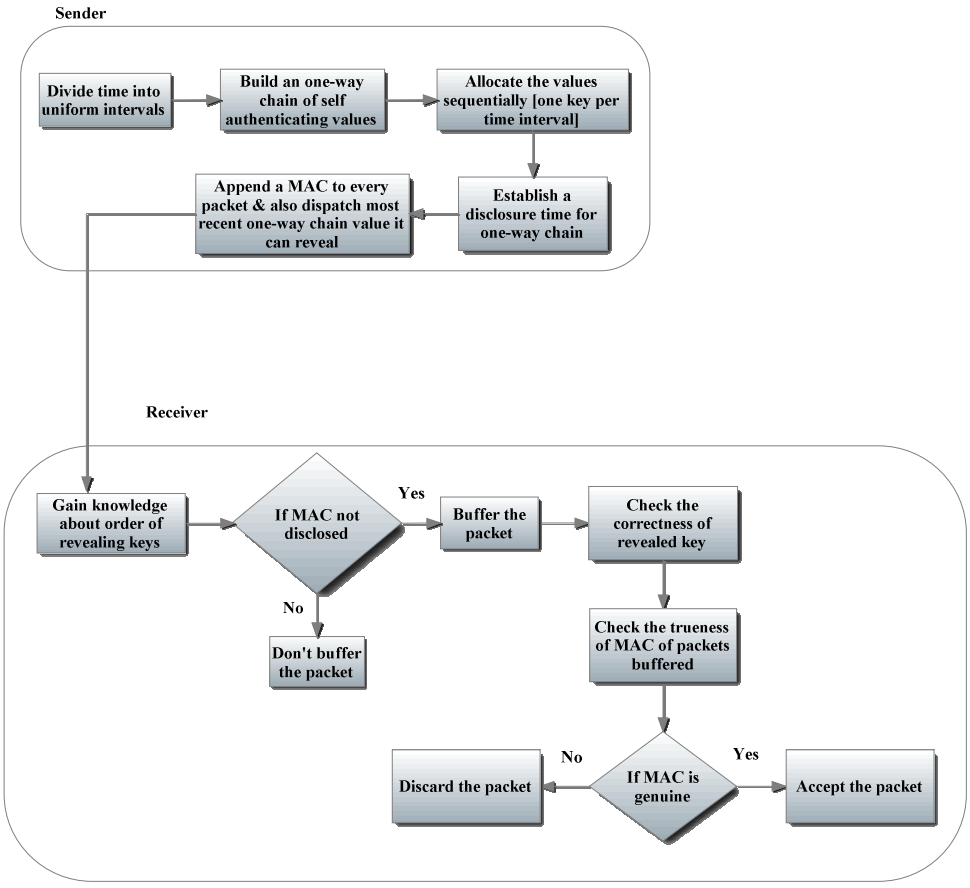


Fig. 3. Working of TESLA protocol.

There are four stages in the basic TESLA protocol. They are: sender setup, bootstrapping the receivers, broadcasting authenticated messages and authentication at receiver.

f. Sender Setup

The time is split into uniform time periods of duration T_{int}. Time period 0 will begin at time $t = T_0$, time period 1 at time $t = T_1 = T_0 + T_{int}$, etc. Since, the one-way chain is utilized in reverse sequence of generation, any entity of a time period can be used to deduce values of previous time periods.

The sender finds the range N of the one-way chain K_0, K_1… K_N and this length bound the highest transmission length before a fresh one-way chain must be constructed. The sender chooses an arbitrary value for K_N. Employing a pseudo-random function f, sender builds the one-way function F: $F(k) = f_k(0)$. The remaining part of the chain is calculated recursively by $K_i = F(K_{i+1})$. This can be generalized as $K_i = F^{N-1}(K_N)$, so we can calculate any entity in the key chain from K_N even if some values are missing.

The construction of the one-way key chain can be generalized in the Fig. 4,

g. Bootstrapping the Receivers

Once the receiver is loosely synchronized in time with sender, it is ready to approve the messages with TESLA protocol. The receiver also needs to cognize about the disclosure arrangement of keys, and get an authenticated key of one-way key chain. The sender sends the key disclosure arrangement by sending the information to the receivers over an authorized channel, which can be done through a digitally signed broadcast message or by unicasting with each receiver. The information to be sent over the authenticated channel is:

- Time interval schedule: The duration of the interval (T_{int}), start time (T_i), index of the interval (i) and the duration of the one-way chain
- The delay (d) of the revealing of the keys
- A key commitment to the key chain K_i, where $i < j-d$ and 'j' is the current period index

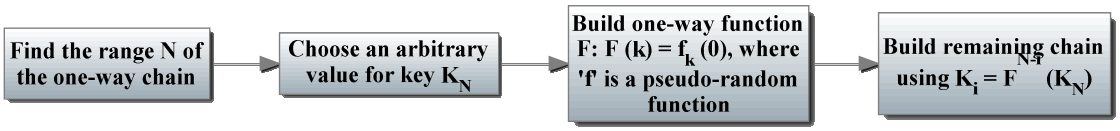


Fig. 4. Construction of one-way key chain.

h. Broadcasting Authenticated Messages

Every key in the one-way chain coheres to a time period. Whenever a sender broadcasts a message, it attaches a MAC to the message with the key corresponding to the time period. The key remains undisclosed for the next d-1 intervals. Therefore messages sent in period 'j' successfully reveal key K_{j-d}, where'd' is the key disclosure delay.

We know that using the similar key many times is not good in various cryptographic operations. So, using key K_j to derive both key K_{j-1} and to calculate MACs is not advised. Utilizing a pseudo-random function f`, the following one-way function can be constructed, F`: $F`(k) = f_k`(1)$. F` is used to derive the key and to calculate the MAC of the messages: $K_i` = F`(K_i)$.

The one-way key chain is obtained utilizing the one-way function F, whereas the derived MAC keys are obtained utilizing the one-way function F`. The broadcasting of the authenticated messages is highlighted in Fig. 5.

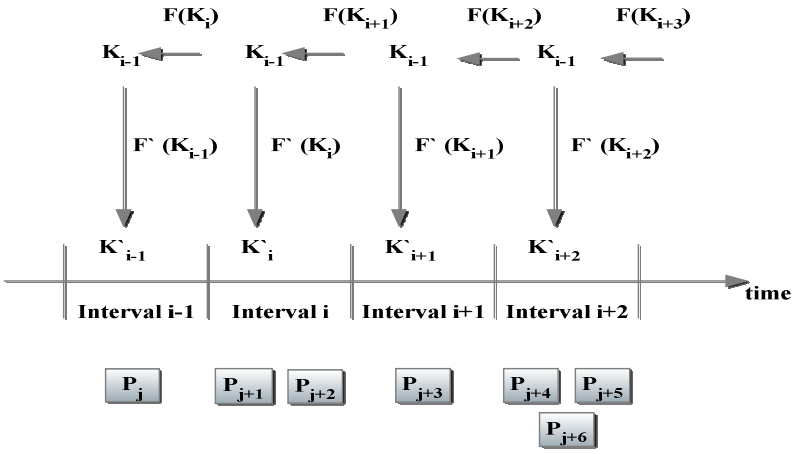


Fig. 5. Broadcasting of authenticated messages.

During the broadcasting of the message M_j in the interval 'i', the sender builds the packet P_j, given in the following notation.

$$Pj = \{Mj||MAC\ (K`i, Mj)||Ki - d\}$$

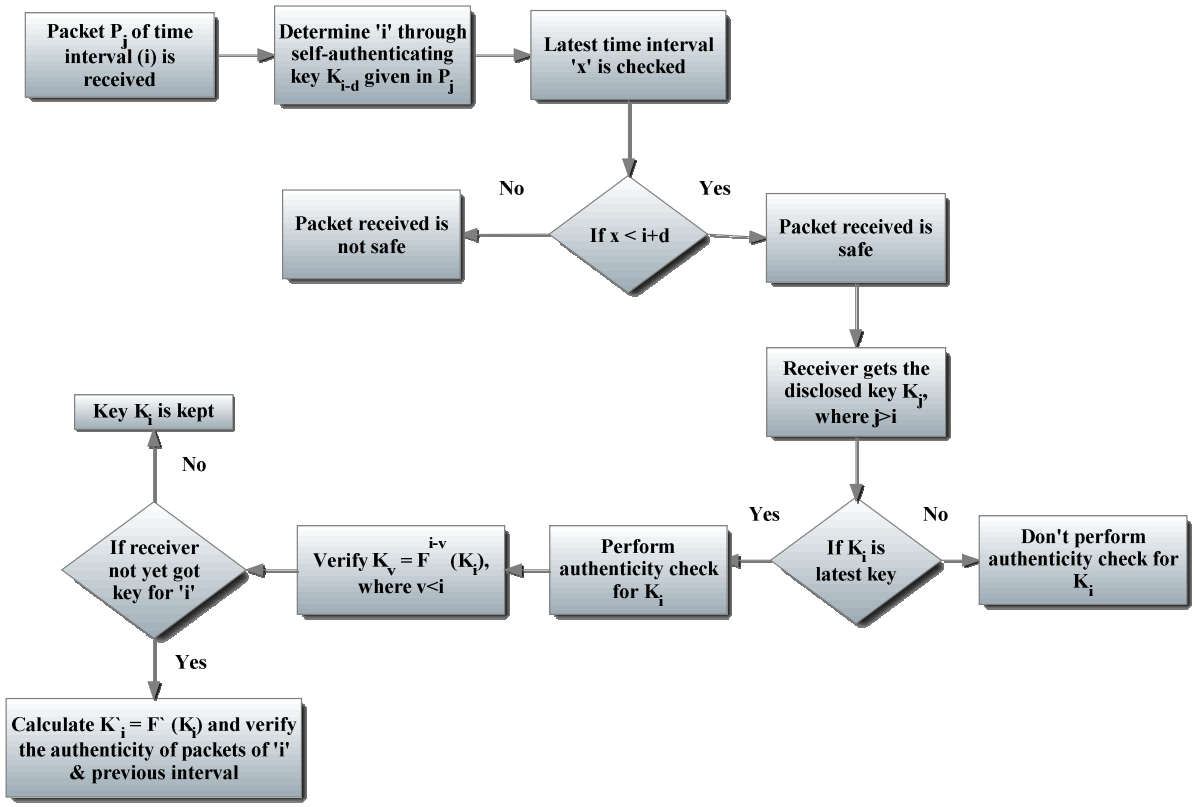
In Fig. 5, time advances from left to right. The time period is split into uniform time intervals. The packets to be sent at each time interval are shown at the bottom of the figure. For every packet, the sender utilizes the key cohering to the MAC of the data through key $K`_{i+1}$. For an assumption of key disclosure delay of two time intervals i.e. d =2, the packet P_{j+3} would also carry key K_{i-1}.

i. Authentication at Receiver

When the sender is revealing a key, every agent has the capability to access that key. A hostile user can create a fake message and move forward a MAC using the revealed key. So as the packets arrive, the receiver must check their MACs. The MACs should be established only on safe keys, i.e. the key is known only by the sender. The packets or the messages which have been computed with those safe keys alone have their respective keys. Receivers must leave away any packets that are not safe, because it may have been altered.

Packet P_j is sent in the time interval 'i'. As the receiver gets the packet P_j, the receiver through the self-authenticating key K_{i-d} revealed in P_j, determines the time interval 'i'. Then, the recent possible time interval 'x', the sender could currently be in, is checked. If $x < i+d$, then the packet received is safe. The sender has not yet attained the interval, where it reveals key K_i, i.e. the key to verify the packet P_j. So, the receiver cannot yet check the trueness of packet P_j sent in time interval 'i'. Instead, the triplet (i, M_j, MAC ($K`_i$, $M_j$) is added to a buffer and that checks the actuality after it cognizes $K`_i$. The security of TESLA protocol does not depend on any grounds on network propagation delay, since each receiver locally finds the safety of the packet. Only if key disclosure delay is not much larger than network propagation delay, the receivers can find that the packets are not secure.

When the receiver gets the disclosed key K_i, it sees if it already knows K_i or any following key K_j, where j>i. If, K_i is the latest key to be received, the receiver verifies the rightfulness of K_i. This can be done by verifying that $K_v = F^{i-v}(K_i)$, for any previous key K_v, where v is less than i. The receiver calculates $K`_i = F`(K_i)$, and checks the rightness of packets of time interval 'i' and of previous time interval, if the receiver has not yet obtained the keys for these intervals. The authentication at the receiver can be summarized in Fig. 6.



Authentication at the receiver.

IV. BACKGROUND WORK

TESLA protocols are being used widely in source authentication for guaranteeing broadcast communication with efficient MACs. The other works relevant to the TESLA protocol are given as follows:

j. TESLA in Controller Area Networks

They are employed in warranting the security of Controller Area Networks (CAN) [12]. TESLA protocol when implemented in CAN bus has a demerit that is crucial for auto-motives. The delay from the TESLA protocol cannot be taken away. The main purpose is to determine the lower limit of the delay. Delays around milliseconds are satisfactory, but such delays do not seem to be less enough for intra-vehicular communication.

k. µTesla Protocol (A Modification of TESLA Protocol)

A further modification of the TESLA protocol is the µTesla protocol [6]. This was proposed to overcome the authentication problem, which guarantees that no hostile users can impersonate the real sender to gain control of the sensors. The protocol attaches 24-bytes MAC to every 30-bytes message, where the MAC is established on symmetric technique but can attain asymmetric property. The protocol is taken for a general case that a sensor can either calculate the MAC to authorize the message or just pass over the MAC to look into the content of the message. This protocol is used where a sensor wants to approve all the messages. In the µTesla protocol, the first message is bootstrapped through an authorized

channel. The µTesla protocol comprises of two phases. For a case of four receivers and one sender, in the first phase, the first key is sent to every receiver in enigma through the secret key known only between the sender and the receiver. During the second phase, the sender divides the time into six intervals and sends out the messages.

First, the bootstrap phase is modified. The whole message is encrypted using a secret key, so that no unauthorized agent can see the message content. The receiver does the encryption and it is confirmed that it is sent from the genuine sender. This gives the authentication property. The message content comprises of keys, related information. An additional CRC (Cyclic Redundancy Check) is used to guarantee that the message encrypted content will not be changed during the transmission. This gives the integrity property. Second, the protocol steps of the broadcast phase are modified. The message is concatenated with its CRC, to secure the integrity property of the contents of the message. Then, encrypt the whole message at various time intervals with the corresponding keys. In the µTesla protocol, every message is authorized it's MAC by calculating one hash function, whereas in C-µTesla protocol, every message is authorized with calculation of one CRC and operation of one symmetric decryption.

l. EKD-Tesla Protocol (An Alteration of µTesla Protocol)

A wireless sensor network comprises many units of small sensor nodes and so these nodes are prone to attacks from hostile agents. The EKD-TESLA (Early Key Disclosure) protocol [26] is a modification of the µTesla protocol, which is more powerful in energy use and resistance to attacks like denial-of-service (DoS). Any authentication protocol should be competent in terms of communication, memory, confidentiality, authentication and computation overhead. µTesla protocol is just a small version of TESLA protocol. In this protocol, every node keeps a secret key (or) master key in the memory, which it shares with the sink. This key is used to encrypt the messages, revive the keys or derivate other keys. When a node needs to send a packet, it calculates the MAC of the message to be sent, by employing a key to the message. This key is known as the MAC key.

The MAC resembles the packet signature. A node receiving a packet and the packet signature will deploy the MAC key and compare the result with the real sent packet. The MAC key is produced from a key chain. The MAC key is deduced from the master key and is produced using a public one-way function. The receiver requires the MAC key to authenticate a packet. The MAC key is generated by a one-way function, by hashing the key deduced from the master key. To generate these keys, the output of the hash is referenced as input of the one-way hash function. This key is not given to the receiver, but instead with a delay following an interval pattern.

A delayed disclosure technique is used in µTesla, which is advanced over using the symmetric mechanism. But this method forces the receivers to delay the packets for at least two time intervals, expecting for the disclosure of the keys in order to authorize those packets. A hostile agent could also attack by sending a bogus message into the network, conscious of the receivers which will have to buffer the messages. The buffering occurs for all the incoming messages. This attack is called as *Denial of Service (DoS)* since the receiver is made to allow all the packets temporarily. In the EKD-TESLA protocol, a sender will generate a MAC key and spread it to its receivers at least one beacon before the MAC is used to authorize the packet. The revealing of the MAC key prior to the sending of the packet that needs to be authorized, allows the nodes that get the packet to immediately authorize the packet and prohibit any DoS as in µTesla.

m. TESLA in Micro-Payment System

The TESLA protocol is being used for the security in micro-payment systems [19] like e-coupons, which can be used by the users through their own devices like laptop, PDA, mobile phone.

The payment system here involves coins/paywords that are vendor-specific. Security needs to be established since the coins could be grabbed in transit and submitted to the vendor in real-time. This not only gives a competent method of source authentication, but also delivers economical security and avoids the *man-in-middle* and DoS attacks. TESLA cannot allocate non-repudiation, i.e. the receiver cannot urge a third party that the message stream arrived from the original source, which is an important aspect for financial transactions. Limited buffering is enough for the sender and the receiver, therefore timely authentication for every packet. The security also depends on the fact that the previous keys become redundant after a time period.

n. TESLA in Multicast Routing Authentication

A multicast protocol [2] allows a sender to efficiently spread the information to many receivers. Multicast authentication is used to protect against the packets injected by hostile users. This is done by enabling a receiver to authorize the packet source and remove the infected packets. TESLA is suitable for providing source authentication for the ALC (Asynchronous Layered Coding) protocol and the MESP (Multicast Encapsulating Security Payload) header. TESLA can be used both in the application layer and in the network layer. In TESLA the receiver never accepts the message as authentic unless it was sent by the real sender. In Advanced Tesla, immediate authentication is employed to levitate the problem of buffering. Receivers with high network delay cannot operate with a small disclosure delay because majority of the packets will break the security the condition and therefore cannot be authenticated. Multiple instances of TESLA with various disclosure delays simultaneously would solve the problem. Each receiver can choose which disclosure delay and therefore which instance to use.

In multicast communication, digitally signed packet involves high overhead, which may be useless for resource-limited devices. The computation and communication overhead can be reduced by signature amortization. And the packet loss can be tolerated by a fault-tolerance coding algorithm. Despite these steps, the signature amortization could not fully avoid pollution attacks. A lightweight and pollution attack resistant multicast authentication protocol (PARM) has been created to fulfill the requirements.

o. Extra Security Enhancements in TESLA

The delay in the TESLA authentication gives a threat to the receivers by flooding attacks [21]. The packets are buffered even if they are inauthentic. To avoid these, some extra precautions are taken. The arriving packets are checked whether they have a valid port number and source IP address for the session. It is ensured that a message is not reissued already received in the session and messages are not significantly larger than the size of the packet in the session. Stronger DoS protection needs both the receivers and the senders arrange additional limitations on the protocol. There can be three options to the basic TESLA: Increasing group authentication, not re-using keys during a time period and shifting buffering to the sender.

Increasing the group authentication needs larger per-packet overhead. In order not to reuse the key, it requires two hashes per packet at both ends and the sender must save or reproduce a

longer hash chain. In TESLA, each MAC key was recursively used for all the packets sent in a time period. If the sender never uses a MAC key more than once, then each key would immediately inform each receiver that the sender of each incoming packet knew the next key along the hash chain. The next key along the hash chain was disclosed only once. Thus, a justifiable receiver strategy would be to leave any incoming packets that a revealed a key seen already. Every packet would save the fill rate of the receiver's buffer and would be revealed by the sender, prohibiting memory flooding attacks.

A key to be received in a later packet for authentication prohibits a receiver from authenticating the final part of a message. Thus, to activate authentication of the final part of a message or of the final message before transmission idleness, the sender requires sending the key with an empty message. A variation in the key disclosure arrangement for a message stream should never be disclosed within the message stream itself. This would bring in vulnerability, because a receiver that did not get the notification of the change would still trust in the old key disclosure schedule.

p. TESLA combined with Quadratic Residues Chain

In TESLA only loose time synchronization is required, i.e. only an upper limit for the time value at the registration server is required [11]. The TESLA protocol is modified to use quadratic residue chains. The protocol works upon time synchronization and squaring function for calculating the one-way chain. This results in broadcast for a longer time as the chain is unbounded.

q. Verification of TESLA in MCMAS-X

MCMAS-X is an extension of the OBDD (Ordered Binary Decision Diagrams) based model [16]. MCMAS stands for "a Model Checker for Multi-Agents Systems". The experimental results of the verification of TESLA in MCMAS-X give the memory occupied for the processor to transfer particular number of packets and the time for the transfer. The authentication properties of the TESLA protocol can be verified efficiently by this tool. There is an oscillatory behavior in the memory occupied as the number of packets increases. The heuristic techniques of the OBDD's contribute to this oscillatory behavior.

r. A Combinational Logic with TESLA Protocol

A security-specialized logic is employed along with the TESLA protocol to enhance the cryptography [17]. The logic used is a combination of a standard epistemic logic and CTL (Computation Tree Logic), known as *Temporal Deductive Logic (TDL)*. This logic is based on a computationally-grounded semantics. A temporal epistemic analysis allows reassuring the TESLA authentication property.

s. TESLA Certificates (A Modification to the TESLA Protocol)

A tradeoff between computation and authentication delay exists in TESLA certificates [3], in order to attain a certificate infrastructure that decreases computational complexity affiliated with certificate verification. A modification to the TESLA protocol provides partial authentication in TESLA certificates. The TESLA certificates are applied to the problem of authentication during handoff. A *certificate authority (CA)* is responsible for producing certificates for the authentic elements of the network. The asymmetry property for authentication makes many applications like Voice over Internet Protocol (VoIP), inherently delay sensitive. In TESLA certificates for the packets to be partially authenticated before the

disclosure of respective authentication key, involve multiple staggered keys in the delayed key disclosure. The incoming packets are placed in the staggered TESLA authentication buffer. When the application considers that the service quality given to the user is not acceptable, it will release the partially authenticated packets in order to increase the delivered quality.

t. DoS attack-tolerant TESLA in Internet of Things

TESLA++ designed for VANET (Vehicular Ad-hoc NETwork), is a DoS-tolerant version [25]. TESLA++ is not suitable for Wireless Sensor Networks (WSN) as it consumes high power. A TESLA based protocol with low consumption of power and tolerant against DoS attack is designed.

Internet of Things (IoT) specifies distinctively recognizable objects (things). They are represented virtually in an Internet-like structure. It is difficult to secure the broadcast communication, because the receivers cannot buffer the data, as they need to process them immediately and the receivers may be dynamic, with elements joining and leaving the network at any time. Also receivers are heterogeneous in computation resources and bandwidth. In TESLA++, only a self-generated MAC is stored to decrease the memory requirements. The sender broadcasts the MAC and then only sends the corresponding key and message, since the receivers store only an abbreviated version of the sender's data. Attacks on storing shortened MACs and broadcasting MACs are reduced without fall in security. µTesla has limited scalability owing to its unicast-based distribution of initial parameter and it cannot levitate DoS attacks.

In Multi-Level µTesla, the initial parameters are predetermined and broadcasted. To further increase the security against DoS attacks, random selection strategies and redundant message transmissions are used with the messages that distribute. Since this protocol needs more memory it is difficult to be implemented in WSN. In WSN, DoS attacks occurs through instances like spoofing, reprogramming attacks, HELLO floods, Synchronize attack, Path based DoS, Replaying and Desynchronization attack. In the modified TESLA protocol, the WSN communicates with the VANET always. The sequence of sending the packets is altered, i.e. the sender first sends the MAC, followed by the message along with the key after time delay. When the size of the message is larger than both MAC and message, the receiver need not store the message.

A two-level key chain composing of high-level key chain and low-key chain is used. The high-level key chain is used to authorize the commitment of every low-level key chain. The high-level key chain has a long time interval to split the time line so that it can cope up the lifespan of a sensor network, without many keys. The low-level key chains have short time intervals, so that the delay and the verification of the messages are acceptable. Computational DoS attack occurs due to broadcasting MACs alone, whereas memory-based DoS attack occurs due to the ability of maximum storage and shortened MAC storage. The attack due to shortened MAC storage can be avoided by relatively small receiver MACs and small time intervals.

µTesla cannot withstand both memory and computational based attack. As the network grows, µTesla needs more space to save longer key chain. It is also more time consuming during the initial step because of its unicast characters.

µTesla and the modified TESLA use Secure Network Encryption Protocol (SNEP) for the initial step of authentication, which depends on pre-sharing the master key and thereby resulting in lower computation consumption.

u. Tesla with Instant Key Disclosure

TESLA protocol is altered with a function of instant key disclosure, known as TIK (Tesla with Instant Key disclosure) [14]. This protocol is counter defense against *wormhole attack*, a strong attack in the ad hoc networks. The attack is possible even if the attacker has not negotiated any hosts and even if all communication provides authenticity and confidentiality. In the wormhole attack, an attacker notes the packets at one location, tunnels them to another location, and resends them into the network. A packet leash is used to detect and defend against wormhole attacks. A leash is any information that is attached to a packet designed to limit the packet's maximum allowed distance. There are two types of leashes. Geographical leashes guarantee that the packet recipient is within a particular distance from the sender. Temporal leashes guarantee that the packet has an upper limit on its lifespan, which limits the maximum travel distance, because the packet can travel at most the velocity of light. Either type of leash can prohibit the wormhole attack, since it allows the packet receiver to detect whether the packet traveled further than the leash allows.

TIK is able to detect a wormhole attack since it implements a temporal hash. It is based on symmetric cryptography. TIK needs accurate time synchronization between all intercommunicating nodes. It also requires each communicating node to know only one public value for each sensor node. An explicit timestamp with an expiry can be attached to each packet for the temporal leash, which makes TIK as an authentication protocol. TESLA has longer time intervals than TIK, to decrease the amount of computation to authorize a new key. TIK has a merit over hop-by-hop authentication with TESLA. In TIK, packets can be verified instantly, since key disclosure always happens in the same packet as the data protected.

v. Testing TESLA with TAME

TAME (Timed Automata Modeling Environment) is an interface to PVS (Prototype Verification System) [1]. It is used to specify and prove the properties of automata. TAME gives a set of proof steps as PVS strategies. Some assumptions are made while modeling TESLA in TAME, like collusion among the hostile users does not lead to additional power, both send time and receive time are calculated on the receiver's clock and the use of pseudo-random function is neglected. Another assumption is that the packet index is part of its content.

w. TESLA Certificate in Hybrid Wireless/Satellite Networks

TESLA Certificates combine the identities of the main elements of their key chains and messages from the senders are authorized by computed MACs [24]. Here, the certificate authority (CA) is the satellite which produces the certificates. It also resembles a proxy for the senders in revealing MAC keys to the receivers in the network. The satellite is chosen as the CA because the satellite is always connected to entire network, physically secure and available. Also satellite has higher storage, renewable energy via solar power and higher computing power. A TESLA permits a user to add authentication to the packets for a single time interval. Therefore, a sender that transmits for multiple times will require many TESLA certificate from the CA. When there are many sources that transmit data over long time intervals, this can pile up to a substantial overhead.

V. DISCUSSION AND RESULTS

The security of TESLA depends on the following assumptions:

- Receiver's clock is synchronized in time up to a maximum error of Δ (maximum time synchronization error)
- Functions F, F` are safe PRFs (Pseudo random functions)

As long as these assumptions are kept, the TESLA protocol is computationally intractable for a hostile attacker to alter a TESLA packet.

The performance of TESLA protocol working on Diffie-Hellman Key Exchange was implemented in NetBeans IDE. In the implementation, first the shortest path is established from a sender node to all other receiver nodes. The secret key is generated using some large prime number. An input file is taken and split into many files. The one-way key chain is generated for the group of files. The MAC codes for all files are also generated. The input data is encrypted. A nonce value and public key are sent from sender node to receiver node. When the received nonce is same for both the users, the file transfer is made.

x. Diffie-Hellman Key Exchange vs. Pre-Distribution Key

The comparison of Diffie-Hellman Key Exchange with Pre-Distribution Key when implemented in TESLA protocol is made in terms of parameters like memory and security. Pre-Distribution key is a method where details of the keys are spread among all nodes before deployment [9].

1) Memory: The memory occupied by the nodes varies for different key distribution schemes because of the difference in system lag and processing time. Diffie-Hellman Key Exchange consumes only less memory when compared to the pre-distribution key. The comparison of memory consumption for these key distributions is given in Fig. 7.

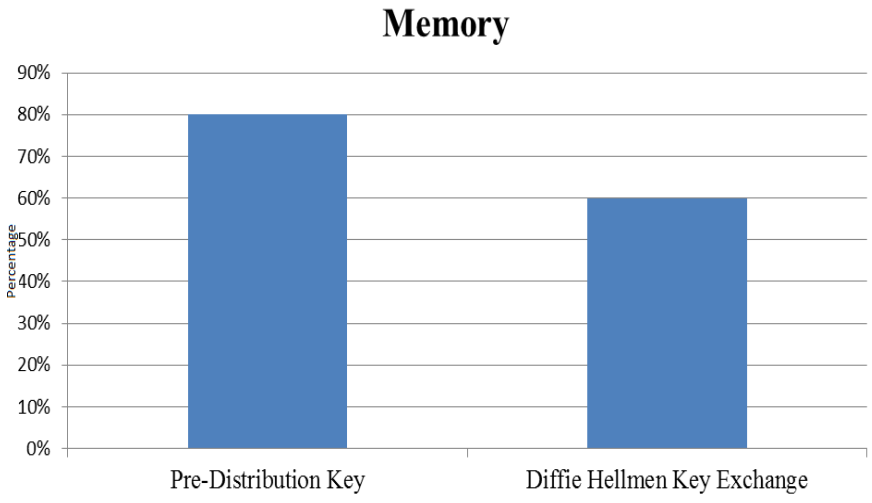


Fig. 6. Diffie-Hellman Key Exchange resulting in less memory usage for TESLA protocol

2) Security: It is the most important aspect in any wireless network, because any unauthorized user can breach the network. The comparison of Diffie-Hellman Key Exchange with Pre-Distribution key in terms of security is given in Fig. 8.

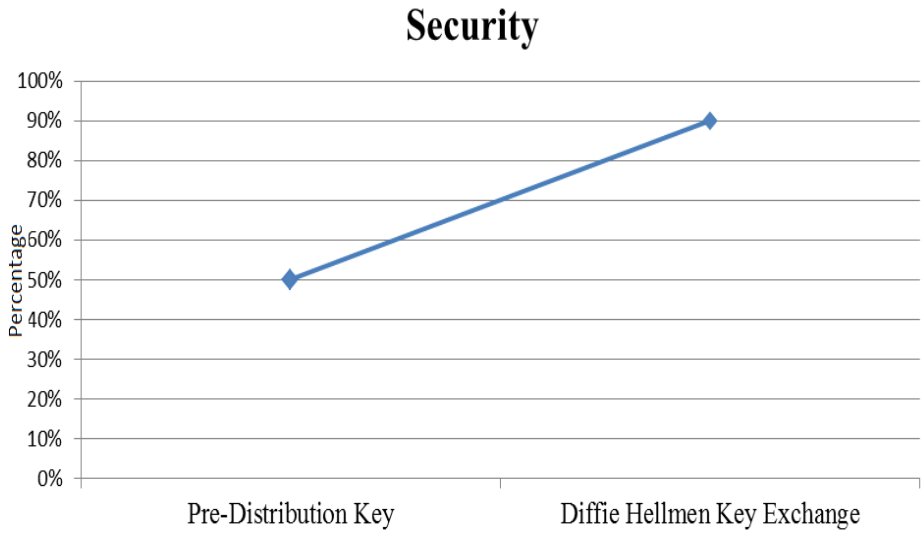


Fig. 7. TESLA protocol with Diffie-Hellman Key Exchange showing enhanced security

y. TESLA protocol vs. CRC Method

The TESLA protocol and CRC method perform differently for various parameters. In the TESLA protocol, the message is split and a MAC is attached for each packet, whereas in the CRC method, instead of MAC, CRC is used for each packets. While some parameters perform well in CRC method, the other parameter performs better in TESLA protocol.

1) Overhead: Higher overhead deteriorates the overall performance of the system. The TESLA protocol shows higher overhead when compared to the CRC method. This is one parameter where the TESLA lags behind the CRC method. The comparison of TESLA protocol with CRC method in terms of overhead is given in Fig. 9.

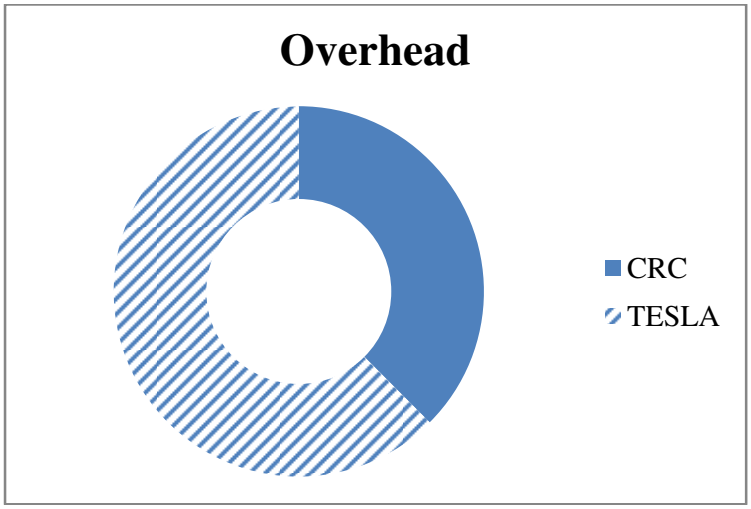


Fig. 8. CRC method exhibiting lesser overhead than TESLA protocol

2) Accuracy: The correctness of the system contributes to the efficiency of the system. TESLA performs better than the CRC method in terms of accuracy. The comparison in terms of accuracy for TESLA protocol with CRC method is given in Fig. 10.

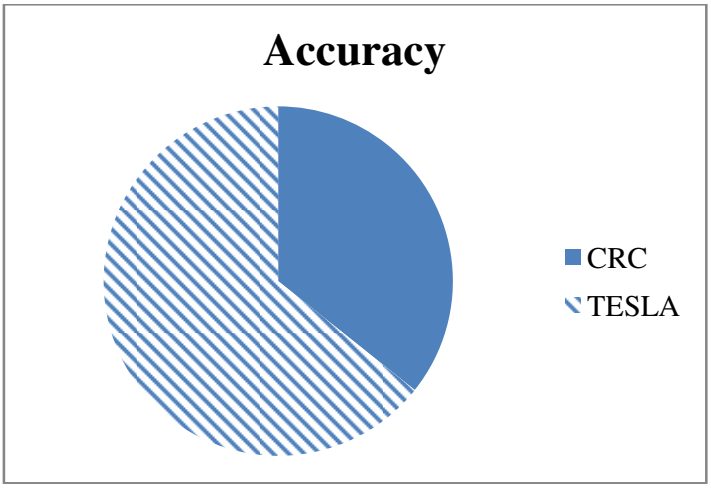


Fig. 9. TESLA protocol providing better accuracy compared to CRC method

VI. CONCLUSION

The security of TESLA protocol can be enhanced in various ways for various applications. In this paper, the secure exchange of the cryptographic keys between the sender and the receiver through Diffie-Hellman Key Exchange is focused. It performs well in aspects like security, accuracy and memory when compared to other techniques. The results prove that the Diffie-Hellman Key Exchange is most suitable for the TESLA protocol for the efficiency of the overall system. Though there are many variants of the TESLA protocol proposed, the TESLA protocol remains as the strong base for all the other protocols, with just few alterations to suit the particular needs of the application.

REFERENCES

1. Archer M (2002), "Proving Correctness of the Basic TESLA Multicast Stream Authentication Protocol with TAME", presented at the WITS '02, Portland, OR.
2. V. G. Babu and D. T. S. Kumar (2011), "Multicast Routing Authentication System using Advanced Tesla", International Journal of Computer Applications, vol. 16, February, pp. 17-25.
3. M. Bohge and W. Trappe (2003), "TESLA Certificates: An Authentication Tool for Networks of Compute-Constrained Devices", in Proceedings of 2003 ACM Workshop on Wireless Security (WiSE `03), San Diego, CA, USA, August.
4. D. Boneh, et al.(2001), "Lower bounds for multicast message authentication," in Advances in Cryptology — EUROCRYPT ’2001, pp. 434–450.
5. R. Canetti, et al.(1999), "Multicast security: A taxonomy and some efficient constructions," in INFOCOMM’99, March, pp. 708–716.

6. W.-H. Chen and Y.-J. Chen (2008), "A C-μTesla Protocol for Sensor Networks," Journal of Informatics & Electronics, vol. 2, March, pp. 29-32.
7. S. Cheung (1997), "An efficient message authentication scheme for link state routing," in 13th Annual Computer Security Applications Conference, pp. 90–98.
8. W. Diffie and M. E. Hellman(2011), "New Directions in Cryptography," IEEE Transactions on Information Theory, vol. T-22, November, pp. 644-654.
9. W. Du, et al.(2003), "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," in CCS'03, Washington, DC, USA, pp. 1-10.
10. R. Gennaro and P. Rohatgi (1997), "How to sign digital streams," in Advances in Cryptology — CRYPTO '97, pp. 180–197.
11. B. Groza, et al., "Broadcast Authentication Protocol with Time Synchronization and Quadratic Residues Chain."
12. B. Groza, et al., "LiBrA-CAN: a Lightweight Broadcast Authentication protocol for Controller Area Networks."
13. N. Haller(1994), "The S/Key one-time password system," in Proceedings of the Symposium on Network and Distributed Systems Security, February, pp. 151–157.
14. Y.-C. Hu, et al.(2003), "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," in IEEE INFOCOM 2003.
15. L. Lamport (1981), "Password authentication with insecure communication," Communications of the ACM, vol. 24, November, pp. 770–772.
16. A. Lomuscio, et al (2006)., "Verication of the Tesla protocol in Mcmas-x," in Proc. CS&P '06, pp. 255-267.
17. A. Lomuscio and B. Wozna(2006), "A complete and decidable security specialised logic and its application to the TESLA protocol," in AAMAS'06, Hakodate, Hokkaido, Japan.
18. S. Miner and J. Staddon (2001), "Graph-based authentication of digital streams," in Proceedings of the IEEE Symposium on Research in Security and Privacy, May, pp. 232–246.
19. V. Patil and R. K. Shyamasundar (2004), "An Efficient, Secure and Delegable Micro-Payment System," in Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04).
20. A. Perrig, et al.(2000), "Efficient authentication and signature of multicast streams over lossy channels," in Proceedings of the IEEE Symposium on Research in Security and Privacy, May, pp. 56–73.
21. A. Perrig, et al (2005)., "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction," June.
22. M. Reiter, "A security architecture for fault-tolerant systems," PhD, Computer Science, Cornell University.
23. M. Reiter, et al.(1994), "A security architecture for fault-tolerant systems," ACM Transactions on Computer Systems, vol. 12, November, pp. 340–371.
24. A. Roy-Chowdhury and J. S. Baras(2008), "A Lightweight Certificate-based Source Authentication Protocol for Group Communication in Hybrid Wireless/Satellite Networks," in Proceedings of IEEE "GLOBECOM" 2008, pp. 1-6.
25. N. Ruan and Y. Hori(2012), "DoS attack-tolerant TESLA-based broadcast authentication protocol in Internet of Things," in International Conference on Selected Topics in Mobile and Wireless Networking, pp. 60-65.

26. P. D. Sene and S. Djanali, "An early key disclosure security protocol for a hierarchical wireless sensor network."
27. D. Song, et al.(2002), "Expander graphs for digital stream authentication and robust overlay networks," in Proceedings of the IEEE Symposium on Research in Security and Privacy, May, pp. 258–270.
28. C. Wong and S. Lam (1998), "Digital signatures for flows and multicasts," in IEEE ICNP '98.
29. Natasa Zivic, "Soft Verification Of Message Authentication Codes" International journal of Electronics and Communication Engineering &Technology (IJECET), Volume 3, Issue 1, 2012, pp. 262 - 285, Published by IAEME.
30. Prof. S.V.M.G.Bavithiraja and Dr.R.Radhakrishnan, "Power Efficient Context-Aware Broadcasting Protocol For Mobile Ad Hoc Network" International journal of Computer Engineering & Technology (IJCET), Volume 3, Issue 1, 2012, pp. 81 - 96, Published by IAEME.
31. Basavaraj S. Mathapati, Siddarama. R. Patil and V. D. Mytri, "Power Control With Energy Efficient And Reliable Routing Mac Protocol For Wireless Sensor Networks" International journal of Computer Engineering & Technology (IJCET), Volume 3, Issue 1, 2012, pp. 223 - 231, Published by IAEME.
32. S. A. Nagtilak and Prof. U.A. Mande, "The Detection Of Routing Misbehavior In Mobile Ad Hoc Networks Using The 2ack Scheme With OLSR Protocol" International journal of Computer Engineering & Technology (IJCET), Volume 1, Issue 1, 2010, pp. 213 - 234, Published by IAEME.