International Conference on Computational Modeling and Security (CMS 2016)

TDHA: A Timestamp Defined Hash Algorithm for Secure Data Dissemination in VANET

Atanu Mondal\textsuperscript{a,*}, Sulata Mitra\textsuperscript{b}

\textsuperscript{a}Camellia Institute of Technology, Kolkata, 700129, India
\textsuperscript{b}Indian Institute of Engineering, Science and Technology, Howrah, 711103, India

\textbf{Abstract}

The safety application in vehicular ad hoc network provides active road safety to avoid road accidents by disseminating life critical information among drivers securely. Such information must be protected from the access of intruder or attacker. A timestamp defined hash algorithm is proposed in the present work for secure data dissemination among vehicles. The sender vehicle sends a deformed version of the original message along with the incomplete message digest to its neighbors. The receiver vehicle generates message digest from the deformed version of the original message and also from the incomplete message digest. It accepts the message if both the digests are equal. The proposed algorithm fulfils all the basic properties such as preimage resistance, collision resistance of a one-way unkeyed hash function. Finally the comparative usability of the hash algorithm in the said application domain is worked out and that shows the dominance of the scheme over the existing schemes.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Organizing Committee of CMS 2016

\textit{Keywords}: Cryptographic hash function; VANET; Secure data dissemination; MD5; SHA-1;

1. Introduction

Vehicular ad hoc network (VANET) receives special attention in terms of traffic security and traffic management. In order to reach the demand of vehicle security, vehicles often broadcast traffic related message among themselves. A security attack on such messages can have severe harmful or fatal consequences to legitimate users. Hence it is prerequisite to achieve security and privacy preservation in real life VANET. It can be achieved by secure data dissemination which maintains confidentiality, integrity and privacy of such messages. Hash algorithms play an important role in modern cryptography. They are widely used in a variety of security applications such as node authentication, message authentication, password protection, digital signature etc. The hash function uses a string of arbitrary length as its input and creates a fixed-length string as output. The fixed-length hash value is often called message digest. The most widely used hash functions are one-way functions for which finding an input which
hashes to a pre-specified hash-value is very difficult. Hash functions may be split into two classes: unkeyed hash functions, whose specification dictates a single input parameter (a message); and keyed hash functions, whose specification dictates two distinct inputs, a message and a secret key. Two commonly used functions are MD5 and SHA-1. Both MD5 and SHA-1 are derived from MD4 which has been known for its weaknesses. MD5 which uses a hash algorithm with 128 bits output has been designed in 1991 and in 2005 it was shown how quickly random collisions for MD5 can be constructed. Also it is not suitable for applications that rely on the properties like secure sockets layer certificates or digital signatures. In [1] authors have shown that how a pair of X.509 certificates can be created that result in the same MD5 hash digest. Then cryptographers began recommending the use of other algorithms, such as SHA-1 which has since been found to be vulnerable [2] as well and most U. S. government applications now require the SHA-2 and SHA-3 family of hash functions [3, 4]. In case of message digest, actual or original message can never be obtained by the receiver; rather the fingerprint of the message is received. So, the use of message digest for data dissemination is not justifiable in VANET. In Message Authentication Code and Hash-based Message Authentication Code a secret key is shared among the sender and receiver. Hence they suffer the secret key exchange problem. Moreover most of these widely used hash functions are useful in large conventional networks. A dynamic privacy-preserving key management scheme for location-based services in VANET is proposed in [5]. It ensures the anonymous authentication of a vehicle and enables double-registration detection. In addition, each vehicle can use a one-way hash function to update the new session key of the vehicles. But the scheme has huge computation overhead for message signature generation and verification. In [6] a vehicle must broadcast a message and establish a common key with receiver vehicles before accessing services from road side unit (RSU). The common key is utilized to guarantee the security while communicating the message. However the common key is established by using identity-based cryptography whose computation complexity is quite great. Moreover, the time of rekey or the time of changing the pseudonym identification of vehicles is not addressed in [6]. The privacy of the drivers, i.e., to abstain the leakage of the real identities and location information of the drivers from any external eavesdropper is preserved in [7]. Each on-board-unit (OBU) is preloaded with a set of anonymous digital certificates, where the OBU has to periodically change its anonymous certificate to mislead attackers [8, 9]. Consequently, a revocation of an OBU results in revoking all the certificates carried by that OBU leading to a large increase in the certificate revocation list size. The ABAKA protocol in [10] uses elliptic curve cryptography at RSU to authenticate requests from multiple vehicles together. It requires a tamper-proof device to be installed in vehicles and requires service providers to generate session keys for connection with vehicles. In [11], the central authority derives a secondary secret key from its private key and securely sends it to the RSU. A RSU is made to sign and deliver messages to the end users on behalf of central authorities. The receiver verifies a message by checking both the correctness of the key signature and the location of the sender. Each vehicle periodically generates a new hash chain and sends it to the central authority. The central authority generates an authentication code from the hash chain and sends it to the vehicle. The authentication codes are used as signatures for messages and RSUs are used for relaying messages between vehicles and central authority. In [12], an efficient privacy preserving data forwarding scheme for service oriented VANGTs is proposed. It relies on encrypting a message by each relaying hop and thus prevents any adversaries from tracing message flows. But both [10, 11, 12] have a lot of computational overhead.

VANET suffers from many constraints due to high mobility of vehicles and use of wireless communication channel. The traditional cryptographic algorithms are not suitable in VANET due to their delay for huge and complex computation. Due to these limitations, it becomes mandatory to devise a security solution for VANET. In the present work, a vehicle generates a message after observing an event within its coverage area generates a message digest from the original message using a timestamp defined hash algorithm (TDHA) and disseminates an incomplete version of the message digest along with a deformed version of the original message among its neighbours. The proposed VANET is a hierarchy having certifying authority (CA) at the root level, base stations (BSs) at the intermediate level and vehicles at the leaf level. Each vehicle has an electronic license plate in which its modified VIN [13] is embedded in encrypted form (E_VIN) by the vehicle manufacturer. The electronic license plate of a vehicle broadcasts (as per IEEE P1069 and IEEE 802.11p) E_VIN after entering into the coverage area of a new BS. The new BS verifies the authentication of the vehicle using an initial registration phase and assigns a digital signature (D_Sig) to the vehicle if it is authentic [13]. The proposed scheme considers the dissemination of infotainment, warning, emergency and beacon messages. The sender vehicle generates a message M and pads a timestamp value with M to generate M’. The timestamp value indicates the exact time instant of generation of M. The sender vehicle generates a deformed version of M’ and an incomplete message digest (IMD) from M’ using
TDHA. It disseminates the deformed version of $M'$ along with the (IMD) among its neighbours. The receiver vehicle computes the message digest (MD$_1$) from the received deformed version of $M'$ using TDHA and also computes the complete message digest (MD$_2$) from the received IMD. The receiver vehicle discards the received deformed version of $M'$ if $MD_1 \neq MD_2$. Otherwise it generates $M'$ from the received deformed message and retrieves $M$ from $M'$ by omitting the timestamp.

Unlike [14] large prime number is not required to generate the message digest. The timestamp field is padded as a key with $M$ to generate $M'$. The key is derived from the system clock during message generation and hence the key value changes from one message to other message. It is not required to store and compute this key. The key is omitted along with the message which helps to reduce the storage and computational overhead for key. Each message contains the digital signature of the sender vehicle to ensure message dissemination among authentic vehicles only. Moreover TDHA is known to the authentic vehicles only. So if any intruder or attacker gets the deformed message and corresponding IMD during transmission then only by brute-force attack [15] they can obtain the original message which makes the proposed TDHA unique from other existing schemes. Unlike [14] the receiver vehicle is able to generate the original message from the received deformed message which is an essential requirement of VANET.

2. Present work

Let the $v_{th}$ vehicle ($V_v$) enters into the coverage area of $B_{th}$ BS (BS$_B$, $1 \leq B \leq NO_OF_BS$ where NO_OF_BS is the total number of BSs under CA) under CA. The number of vehicles under BS$_B$ is NO_OF_V$_B$. The electronic license plate of $V_v$ broadcasts E VIN (EVIN$_v$), BS$_B$ receives EVIN$_v$, verifies validity of EVIN$_v$, generates a digital signature ($D$ Sig$_v$) if E VIN$_v$ is valid and assigns $D$ Sig$_v$ to $V_v$ [13]. $V_v$ selects the available options at OBU in preferable language to generate the message M after observing an event. The interface of OBU is represented by UNICODE-88. It converts the options that are selected by $V_v$ from the OBU into binary format. The message M is in the form of ($D$ Sig$_v$, Type, MC, L$_v$, Lane$_v$). The size of $D$ Sig$_v$ is 160 bits [13]. The type field in the message format indicates the type of the message. The size of this field is assumed as 3 bits so that the proposed scheme is able to support 8 different types of messages. The MC field in the message format indicates the events or infotainment for which M is generated by $V_v$. The size of this field is assumed as 6 bits to accommodate maximum number of events or infotainment in VANET during simulation. L$_v$ is the location identification of $V_v$ in $k^{th}$ lane identified as Lane$_v$ under BS$_B$ in the message format. The size of the location identification field is assumed as 12 bits and the size of the lane identification field is assumed as 11 bits by considering the transport system of Kolkata [16] as a benchmark during simulation. Hence the size of M is 192 bits. $V_v$ generates $M'$ by padding the timestamp in the form (HH.MM.SS) with M. The highest digit which appears in (HH.MM.SS) is 9 and hence each digit of the time (H, M, S) is represented by 4 bits. Each delimiter character is also represented by 4 bits to support different types of delimiter characters like (,, -, /, ::, |) etc. Hence the size of timestamp field is 32 bits and the size of $M'$ is (192 + 32) 224 bits. In this section the proposed timestamp defined hashing scheme along with the algorithm for generating a short length, fixed hash digest from M is elaborated.

2.1 Timestamp defined hashing scheme

The OBU of $V_v$ generates 192 bits M from the options selected by $V_v$ at the time of message generation. The steps of operation at $V_v$ to generate IMD from M are elaborated as follows:

Step-1: Pads 32 bits timestamp in the least significant position of M of size 192 bits to obtain M' of size 224 bits

Step-2: Divides $M'$ into 14 blocks, each of size 16 bits

$[(B_{11}, B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{17}, B_{18}, B_{19}, B_{20}, B_{21}, B_{22}, B_{23}, B_{24})]

Step-3: Divides $i^{th}$ block into two parts, each of size 8 bits

$[(B_{11}, B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{17}, B_{18}), (B_{19}, B_{20}, B_{21}, B_{22}, B_{23}, B_{24}, B_{25}, B_{26})]

Step-4: Divides 14 blocks into two segments, each having 8 blocks

Segment 1: $[(B_{11}, B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{17}, B_{18}), (B_{19}, B_{20}, B_{21}, B_{22}, B_{23}, B_{24}, B_{25}, B_{26})]

Segment 2: $[(B_{11}, B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{17}, B_{18})]

Step-5: Swaps $B_{11}$R with $B_{12}$R, $B_{13}$R, $B_{14}$R, $B_{15}$R, $B_{16}$R, $B_{17}$R, $B_{18}$R with $B_{19}$R, $B_{20}$R, $B_{21}$R, $B_{22}$R, $B_{23}$R, $B_{24}$R, $B_{25}$R, $B_{26}$R with $B_{27}$R, $B_{28}$R, $B_{29}$R, $B_{30}$R, $B_{31}$R, $B_{32}$R, $B_{33}$R, $B_{34}$R.
Segment 1: \[(B_{13L}, B_{13R}), (B_{12L}, B_{12R}), (B_{11L}, B_{11R}), (B_{10L}, B_{10R}), (B_{9L}, B_{9R}), (B_{7L}, B_{7R})\]
Segment 2: \[(B_{6L}, B_{6R}), (B_{5L}, B_{5R}), (B_{4L}, B_{4R}), (B_{3L}, B_{3R}), (B_{2L}, B_{2R}), (B_{1L}, B_{1R}), (B_{0L}, B_{0R})\]

**Step-6:** Performs XOR operation among the two parts of each block and replaces the block by the result of XOR operation of size 8 bits

\[B_{i} = B_{iL} \oplus X R O B_{iR}\] where \(i\) is the block number

**Step-7:** Replaces \(B_{i}\) by (complement of (\(B_{i}\))) for \(i = 13\) and by (complement of (complement of (\(B_{i+1}\)) \(X R O B_{i}\))) for \(0 \leq i \leq 12\)

**Step-8:** Reduces total number of blocks to 8

- Segment 1: \([B_{L3}, (B_{L6}), (B_{L5}), (B_{L4})]\)
- Segment 2: \([B_{L3}, (B_{L2}), (B_{L1}), (B_{L0})]\)

**Step-9:** Recalculates value of the blocks in Segment 2

\(V_{r}\) transmits the output of Step-5 as deformed \(M'\) along with the 64 bits output of Step-9 as IMD. The receiver vehicle generates the complete message digest from the received IMD by recalculating the value of the blocks in Segment 2 in Step-10 of the scheme.

### 2.2 Timestamp defined hashing algorithm

In this section the algorithm to generate the deformed \(M'\) along with the IMD at the sender vehicle \((V_{s})\) and to generate the complete message digest from IMD at the receiver vehicle are elaborated.

---

**Algorithm at sender vehicle**

**Input:** Message \((M)\) of size 192 bits, Timestamp of size 32 bits;

**Output:** IMD of size 64 bits

```
Begin
    //Step-1[line1 - line2]
    1. Apply padding of 32 bits Timestamp to 192 bits to obtain \(M'\)
    2. \(i \leftarrow 0, j \leftarrow -1\)
    3. while \((i < 224)\) { 
        //Step 2[line 3 – line 6]
        4. \(j \leftarrow j + 1, B_{ij} \leftarrow M'(i, i + 15)\) // Split into 16 bits block
        5. \(i \leftarrow i + 16\)
        6. \(i \leftarrow 0\)
    7. while(\(i \leq j\)) { 
        //Step 3[line 7 - line10] 
        8. \(B_{L1} \leftarrow B(0,7)\) // Left subblock of \(B_{i}\)
        9. \(B_{R} \leftarrow B(8,15)\) // Right subblock of \(B_{i}\)
        10. \(i \leftarrow i + 1\) //Step 4[line11 – line 13]
        11. \(B_{L1} \leftarrow \{B_{13L}, B_{13R}, \ldots, B_{7L}, B_{7R}\}\) // Left segment
        12. \(B_{R} \leftarrow \{B_{6L}, B_{6R}, \ldots, B_{0L}, B_{0R}\}\) // Right segment
        13. \(i \leftarrow 0\)
        14. while(\(i \leq j\)) { 
            //Step 5 [line 14 – line 16]
            15. \(\text{swap}(B_{R}, B_{R}), i \leftarrow i + 1, j \leftarrow j - 1\)
        16. \(i \leftarrow -1\)
        17. while(\(i < 14\)) { 
            //Step 6[line 17 – line 18]
            18. \(i \leftarrow i + 1, B_{L} \leftarrow B_{L} \oplus B_{R}\)
        19. while(\(i \leq 0\)) { 
            // Step 7[line 19 – line 24]
            20. \(i \leftarrow 13\)
        21. \(B_{i} \leftarrow \bar{B}_{i}\)
        22. else \(B_{i} \leftarrow \bar{B}_{i} + 1 \oplus B_{i}\)
        23. \(i \leftarrow i + 1\)
        24. \(i \leftarrow i + 1, j \leftarrow 0\)
        25. while(\(i \leq 7\)) { 
            // Step 8[line 25 – line 32]
            26. \(i \leftarrow 3\)
        27. \(B_{L} \leftarrow B_{j}\)
        28. else if(\(i \leq 7\)) { 
            \(j \leftarrow j - 2, B_{L} \leftarrow B_{j+1}\)
        29. \(i \leftarrow 0\)
        30. \(j \leftarrow j + 2\)
        31. \(\text{BL}_{i} \leftarrow B_{j} \oplus B_{j+1}\)}
        32. \(i \leftarrow i + 1, j \leftarrow j + 2\)
        33. \(\text{BL}_{3} \leftarrow B_{L} \oplus B_{L-2}\) // Step 9[line 33 – line 36]
        34. \(\text{BL}_{2} \leftarrow B_{L} \oplus B_{L-1}\]
        35. \(\text{BL}_{1} \leftarrow B_{L}\)
        36. \(\text{BL}_{0} \leftarrow B_{L}\)
        37. \(\text{End}\)
```

---

**Algorithm at the receiver vehicle**

**Input:** IMD, \(B_{L}\), where \(i = (0, 1, \ldots, 7)\), of size 64 bits; **Output:** Complete message digest of size 64 bits

```
Begin
    //Step-10[line 1 – line 4]
    1. \(\text{BL}_{3} \leftarrow \text{BL}_{1}\)
    2. \(\text{BL}_{2} \leftarrow \text{BL}_{3} \oplus \text{BL}_{1}\)
    3. \(\text{BL}_{1} \leftarrow \text{BL}_{0}\)
    4. \(\text{BL}_{0} \leftarrow \text{BL}_{2} \oplus \text{BL}_{0}\)
    5. \(\text{End}\)
```
3. Performance Analysis

In this section the strength of the proposed hash algorithm (TDHA) is evaluated. The performance of the scheme is studied qualitatively and quantitatively.

3.1. Strength of TDHA

The strength of the algorithm is discussed by showing the extent of maintaining the following basic cryptographic properties of a one-way hash function. Breaking of the proposed hashing scheme requires breaking these cryptographic properties.

i. Preimage resistance - It is computationally infeasible to find any input which hashes to a predefined output i.e. it is difficult to find an input (message) A such that h(A) = H where h(A) is the hash digest of A. It represents the one-way property of hash function.

ii. Collision resistance - It is a property of cryptographic hash function. A hash function h is collision resistant if it is hard to find two distinct inputs which hash to the same output i.e. two inputs A and B such that h(A) = h(B) and A≠B.

iii. Second preimage resistance - Given an input A it should be difficult to find another input B such that A≠B and h(A)=h(B).

Preimage resistance: The first 160 bits of each M’ contains the digital signature of a vehicle which varies from vehicle to vehicle. The last 32 bits of each M’ contains the timestamp which varies from M’ to M’ as it is difficult for a vehicle to generate two messages at the same timestamp. Hence two different M’ generated by the same vehicle never be identical due to their different timestamp value. Moreover two different M’ generated by the two different vehicles never be identical due to their different digital signature and timestamp value. So the 224 bits input message of TDHA is always unique. Hence the output of XOR operations from Step-6 to Step-9 of TDHA is unique for each and every input message of TDHA. A different deformed message and IMD have been generated for each and every M’. Thus, it is computationally infeasible to find any preimage for the proposed algorithm. In the present work it is assumed that TDHA is not known to the attacker or intruder. Hence the only method to generate the original message from the received 288 bits by the attacker is brute-force attack [15]. The attacker assumes an input of size N (N≥1) bits and hence the number of possible input combinations is $2^N$. It generates $2^{288}$ output combinations from each input combination using brute-force operations and one of the output combinations out of $2^{288}$ output combinations is the received 288 bits of the attacker from the network. Hence the total number of operations need to be performed by the attacker is order of $2^N×2^{288}$.

Collision resistance: In TDHA, to find out two different messages A and B with hash digests h₁ and h₂ respectively such that h₁=h₂, the followings need to be satisfied.

If A and B are generated by the same vehicles then they have different timestamp value. If A and B are generated by the different vehicles then they have different digital signature value and timestamp value. So the output of Step-5 which is the input of Step-6 is always different for A and B which in turn causes the output of Step-6 to Step-8 of TDHA is different for A and B. Hence, it is proved that TDHA is collision resistant and the difficulty of coming up with two messages of size 224 bits having same IMD by the attacker starting from the N bits input is the order of $2^N × 2^{224}$ operations by using the brute-force attack [15].

Second preimage resistance: It is known as weaker or easier version of collision resistance. Hence, a collision resistance function is also a second preimage resistant and it needs to perform order of $2^N × 2^{224}$ operations to have same IMD by using brute-force attack [15].

3.2. Qualitative performance

In this section the qualitative performance of TDHA is studied in terms of communication, storage and computation overhead. The qualitative performance of TDHA is also compared with two commonly used hash algorithms viz. MD5 and SHA-1. It is also compared with LOCHA [14]. The result of comparison is shown in Table 1.

Communication overhead: The sender vehicle sends deformed M’ of size 224 bits and IMD of size 64 bits. Hence the communication overhead of TDHA is 288 bits. The communication overhead of TDHA is higher than MD5,
SHA-1 and LOCHA as observed from Table 1. Unlike MD5, SHA-1 and LOCHA the sender vehicle transmits a deformed version of the original message along with an IMD in TDHA so that the receiver vehicle is able to generate the original message from its deformed version which causes an increase in communication overhead of TDHA.

Storage overhead: The operations which are mentioned from Step-1 to Step-4 of TDHA in section 2.3 can be executed in the main memory of the system whereas the operations like swap, XOR, NOT from Step-5 to Step-10 of TDHA need internal CPU registers. The maximum number of operands for swap, XOR and NOT operations is 3 of size 8 bits. The maximum size of operands for other operations like assignment, increment, decrement etc. is also 8 bits. Hence the storage overhead of TDHA is 3 registers of size 8 bits.

Computation overhead: The computation overhead of TDHA is calculated in terms of the number of clock cycles (cc) that are required to execute all the 10 steps of TDHA at sender and receiver. It is 159 clock cycles as observed during simulation.

Table 1. Comparative performance.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication overhead (bits)</th>
<th>Computation overhead (cc)</th>
<th>Storage overhead (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>128</td>
<td>36360</td>
<td>12 registers of 32 bits</td>
</tr>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>84272</td>
<td>12 registers of 32 bits</td>
</tr>
<tr>
<td>LOCHA</td>
<td>96</td>
<td>2952</td>
<td>4 registers of 16 bits, 18 registers of 8 bits</td>
</tr>
<tr>
<td>TDHA</td>
<td>288</td>
<td>159</td>
<td>3 registers of 8 bits</td>
</tr>
</tbody>
</table>

Energy requirement of all the competing schemes for communication and computation: As per NI PXIe - 4339 [17] specification, energy consumption for transmitting 1 byte of data for 8 channel bridge input module by using 24 bits delta-sigma analog-to-digital converter is calculated as follows.

Minimum 10 mV is required to transmit 3 bytes of data in NI PXIe - 4339. Now, 1 V is equivalent to $1.602 \times 10^{-19}$ joule, 10 mV is equivalent to $1.602176565 \times 10^{-21}$ joule and so $1.602176565 \times 10^{-15}$ μJ is required to transmit 3 bytes of data. Hence, the required energy for transmission of 1 byte of data is $0.534 \times 10^{-15}$ μJ.

The different excitation voltage values such as 0.625V, 1V, 1.5V, 2V, 2.5V, 2.75V, 3.3V, 5V, 7.5V, and 10V in NI PXIe – 4339 are used for quarter, half and full bridge configuration mode of data transmission and reception. In the present work, the excitation voltage value is considered as 10V to use full bridge configuration mode for simultaneous transmission and reception of data. Again, 0.22 mV is required for a single clock cycle to send the bit values of the deformed M' and IMD through channel when excitation voltage value is 10V. Since, 1 V is equivalent to $1.602176565 \times 10^{-19}$ joule, 0.22 mV is equivalent to $0.3524788443 \times 10^{-22}$ joule. The energy required per clock cycle is $0.3524788443 \times 10^{-16}$ μJ. Hence the energy required for communication in MD5, SHA-1, LOCHA and TDHA is $8.544 \times 10^{-15}$ μJ, $10.68 \times 10^{-15}$ μJ, $6.406 \times 10^{-15}$ μJ and $19.224 \times 10^{-15}$ μJ respectively. The energy required for computation in MD5, SHA-1, LOCHA and TDHA is $1.2815 \times 10^{-12}$ μJ, $2.97033 \times 10^{-12}$ μJ, $1.04049 \times 10^{-13}$ μJ and $5.6042 \times 10^{-13}$ μJ.

3.3. Quantitative performance

The simulation experiment is conducted in LaneA under BS_B to study the quantitative performance of the proposed scheme in terms of message transmission delay (MTD) and traffic overhead (TOH). In this section the quantitative performance of TDHA is elaborated and compared with the existing schemes [18, 19].

$V_v$ generates the deformed message and the IMD after observing all the events that occurs within its coverage area in LaneA under BS_B. The message transmission delay of $V_v$ (MTD_v) is the sum of the delay due to its all generated messages. The message transmission delay of $V_v$ per message is computed as the sum of the computation time of the deformed message from the original message (Step-1 to Step-5 of TDHA), the computation time of the IMD from the deformed message (Step-6 to Step-9 of TDHA) and the transmission time of deformed message along with the IMD to its neighbour vehicles. So the message transmission delay at BS_B (MTD) is $\sum_{v=1}^{N_{O.F.D.F.B}} MTD_v$ sec.

The traffic overhead of $V_v$ (TOH_v) is the rate of transmission of its generated messages and their digests in bits/sec. So, the traffic overhead at BS_B (TOH) is $\sum_{v=1}^{N_{O.F.D.F.B}} TOH_v$ bits/sec.
Fig. 1 shows the plot of MTD vs. simulation time when the number of vehicles under BS is 150. The rate of occurrence of event is 2 in Fig. 1(i) and 15 in Fig. 1(ii). Fig 2 shows the plot of MTD vs. number of vehicles. The simulation time and rate of occurrence of event are 10 sec and 2 in Fig. 2(i) whereas 40 sec and 15 in Fig. 2(ii).

Fig. 3 shows the plot of TOH vs. number of vehicles. The simulation time and rate of occurrence of event are 10 sec and 2 for Fig. 3(i) whereas 40 sec and 15 in Fig. 3(ii). Fig. 4 shows the plot of MTD vs. number of messages per sec and Fig. 5 shows the plot of TOH vs. number of messages per sec considering the simulation time as 10 sec, number of vehicles as 150 and rate of occurrence of events as 2.

Discussion of results: The number of events in Lane may increase with the simulation time. The number of messages generated in Lane increases with the number of vehicles as well as with the number of events in Lane. Hence MTD of TDHA increases with simulation time, number of vehicles and rate of occurrence of events as observed from Fig. 1 and Fig. 2 whereas TOH of TDHA increases with the number of vehicle and rate of occurrence.
of events as observed from Fig.3. Both MTD and TOH increase with the number of messages per sec as observed from Fig.4 and Fig.5 respectively. In [18] MTD includes the time to collect the segments of the master key by the sender vehicle from its neighbouring vehicles for message encryption and time to verify 1024 bits RSA signature for message authentication during message transmission. In [19] MTD includes the time of transmitting hello message by a vehicle to the RSU before starting a new session of data transmission. It also includes the time of sharing keys between user and RSU for data encryption and decryption after initiating the session. TOH of [19] includes the overhead of transmitting the hello message and sharing the keys. Unlike [18, 19] MTD of TDHA includes its execution time and TOH of TDHA includes message dissemination overhead by different vehicles under BS0 only. It has no key exchange overhead. Hence its performance is better than [18, 19] as observed from Fig.1 to Fig.5.

4. Conclusion

A secure data dissemination among vehicles in VANET is proposed in the present work. Its performance outperforms the existing schemes both qualitatively and quantitatively. The operational activity of TDHA can be made more complicated to make the data dissemination more secure. TDHA can be implemented for the distributed VANET to accommodate more number of vehicles as well as for more number of occurrences of events. The performance of TDHA can also be studied for successful message transmission rate, the rate of drop of messages with respect to simulation time.

References