Trusted Routing for VANET

Terence Chen
National ICT Australia
Locked Bag 9013, Alexandria
NSW 1435, Australia, and
University of New South
Wales, Australia,
terence.chen@nicta.com.au

Olivier Mehani
National ICT Australia, and
University of New South
Wales, Australia, and
Mines ParisTech, CAOR — Centre de Robotique,
Mathématiques et Systémes, France
olivier.mehani@nicta.com.au

Roksana Boreli
National ICT Australia
Locked Bag 9013, Alexandria
NSW 1435, Australia, and
University of New South
Wales, Australia,
roksana.boreli@nicta.com.au

***Abstract*—Trust establishment in VANET is a particularly challenging task due to the lack of infrastructure, openness of wireless links and the usually highly dynamic network topology. To overcome these difficulties, we propose a trusted routing framework that provides message authentication, node-to-node trust and routability verification, without online assistance of Certificate Authorities (CA). Our approach prevents identity impersonation, false link availability indication by compromised nodes as well as some of routing protocol specific misbehaviours. By applying this framework to the Optimised Link State Routing Protocol (OLSR), we demonstrate how this mechanism can be used to establish trusted routes.**

***Keywords*: trust establishment, VANET, authentication, OLSR**

***Area of Interests*: trusted routing, authentication mechanisms**

I. Introduction

Current routing protocols used in Vehicular Ad hoc Networks(VANET) such as OLSR [1], Ad hoc On Demand Distance Vector (AODV) [2] and Dynamic Source Routing(DSR) [3] are designed without considering security. Hence they are susceptible to a wide range of attacks and misbehaviours. Amongst the network security attributes, authentication is particularly important for ad hoc networks due to their cooperative nature. Restricting unauthorised node access to the network is not sufficient, as legitimate nodes may behave abnormally when compromised or hijacked (*e.g.* if the secret keys necessary to generate legitimate messages are captured by an attacker). So the issue to determine could be not only *"is the message's originator authentic?"* but also *"is the message authentic?"*.

A number of trust establishment mechanisms have been proposed as extensions of Mobile Ad Hoc Network (MANET) routing protocols or as stand-alone secure routing protocols. Schemes proposed for distributed authenticated routing [4], [5] and trust-group-based authentication service [6] use distributed authentication authorities and distributed trusted groups to assist the authentication process. However these proposals do not fit in well in the highly dynamic VANET environment, as multi-party authentication introduces high delay. Some secure routing protocols or protocol extensions provide message authentication using different digital signature schemes, such as OLSR extensions proposed in [7], [8], SAODV [9], Secure Link State Protocol (SLSP) [10] and the secure on-demand routing protocol Ariadne [11]. Securing control messages using digital signatures enables the receiver to verify sender's identity and message integrity, but it cannot prevent a compromised node from tampering with the network. The advanced signature system for OLSR proposed in [12] uses link atomic information of surrounding nodes to prevent link spoofing. However it is based on the assumption that all intermediate nodes are authentic, which is not always the case.

In this paper we focus on designing a distributed trusted routing framework that achieves authentication of messages, nodes and routes. Our main contributions are in enabling verification of routes, rather than just securing routing protocol messages or authenticating individual nodes (which is conceptually similar to [12]) and in proposing a new framework for trusted routing. The architecture is distributed and uses limited assistance from a Certificate Authority (CA) which may only be used for initial distribution of keys and certificates, or for infrequent certificate updates. As an example, we have applied the framework to OLSR and evaluated the performance and overhead introduced by the trust related extensions. We note that the generic mechanism is also applicable to several other VANET routing protocols.

The paper is structured in the following way. we explain the components of the trusted routing framework in Section II. In Section III we apply the framework as an extension to OLSR. Performance evaluation in regards to resilience to attacks and trust related overhead is presented in Section IV.

II. The Proposed Framework

The trust establishment framework consists of three modules which are designed to address different threats in the network. Digital signature are employed to assure the integrity and authorship of the control messages. The Node-to-node authentication module, implemented as a light-weight version of Secure Neighbour Discovery (SEND) [13] mechanism, enables network entities to identify each other quickly in a minimum number of iterations. The Cumulative Routability Verification module prevents compromised nodes from declaring false link information. By collecting sufficient evidence, each node is able to verify all routes without additional actions needed outside of the standard routing protocol.

A. Prerequisites

A Public key cryptosystem is used for digital signatures and secret exchange in our framework. An off-line CA issues keys and certificates to all participants before they form a network. Nodes can renew their keys and certificates when they are in contact with the CA server. Each node is required to have the following components:

- A public/private key pair. *i.e.* K_i^+/K_i^-. The private key remains secret to other nodes.
- The public key of CA, *i.e.* K_{ca}^+.
- A certificate signed by the CA, which binds the network ID/IP address to the node's public key. The certificate also includes a valid time and expiration time. A new certificate will be re-issued before this expires.
- All nodes are loosely synchronised, using the network time protocol or a GPS device.

Instead of pre-storing all other nodes's public keys, or obtaining the authenticating node's certificate from a central server, the certificate is pre-distributed and provided by the corresponding node at the time of authentication. The certificate has the following format: $Cert_i = [ID_i, K_i.^+, T_v, T_e]_{K_{ca}^-}$ where T_v is the validity time and T_e is the expiration time.

B. Message Authentication

This module specifies the way to protect control messages using digital signatures. A *digital signature* of a message is a value dependent on a secret known only to the signer, and additionally, on the content of the message being signed. Usually a smaller, fixed-size message digest, generated by a hash function, is signed instead of the complete message. In public key cryptosystems, the sender signs the message digest using its private key and receivers can verify the authorship of the message with signer's public key. The one-way property of hash functions also assures the message has not been modified.

The sender must attach a digital signature at the end of every control message. Note that some routing protocols use variable fields in their control messages, such as hop-count and time-to-live field. In this case, these fields should be excluded from the digital signature.

C. Node-to-Node Authentication

The first line of defense against attackers is identity authentication. A legitimate device will be given a pair of public/private key in a secure way before communications, *e.g.* the keys may be entered manually or through secure transfer protocols. Assuming the private key is only known to the designated node and stored in a perfectly secure way, proving a node has the corresponding key is equivalent to proving a node's identity. The node-to-node authentication module defines the identity authentication handshake between two nodes. To address the replay and man-in-the-middle attacks, two nodes challenge each other with random values/nonces.

During the authentication procedure, the node attempting to authenticate presents its identity and certificate to the authenticating node, as we would present an ID card to prove our identity in real-life. The authenticating node will first verify the certificate using the public key of the CA and then challenge the initiating node by encrypting a nonce with the initiating node's public key, to test whether it has the corresponding private key. At the end of the handshake, two nodes exchange secret keys (encrypted with other's public key) for quick re-association in the future.

The authentication procedure is functionally divided into three phases (including certificate/key distribution and re-association) as shown in Figure 1.

1) step 1-2: Public/private key pairs and certificates are distributed to legitimate nodes who wish to join the ad hoc network. This phase can be done any time before two nodes try to authenticate each other.
2) step 3-7: In the second phase two nodes exchange certificates, verify identity of each other by sending challenges, and in the last two steps they exchange secrets for re-authentications in the future. A loose timestamp is included to prevent replay attack.
3) step 8-9: When two nodes happen to be disconnected for a period of time and try to re-authenticate with each other, they will make use of pre-share secrets exchanged in step 6 and 7. This phase allows two nodes to recognise each other quickly without going through all the full authentication steps as in phase 2.

D. Cumulative Routability Verification

The basic idea of the routability verification mechanism is that a node must provide a piece of evidence from a neighbour node proving that it can route to this neighbour. The evidence must be able to prove the relationship between two nodes, and no other entity should be able to reproduce the same information (including the message originator). In a public-key cryptosystem, digital signature of link status information from the neighbour nodes satisfies these requirements. After two nodes have verified each other's identity using node-to-node authentication mechanism, they exchange a Routability Certificate (RC) in the format of $[ID_A, ID_B, Cert_B, [ID_A + T_1]_{K_B^-}]$. The certificate can be read as "node A claims that it can route to node B, and node B approves the information that

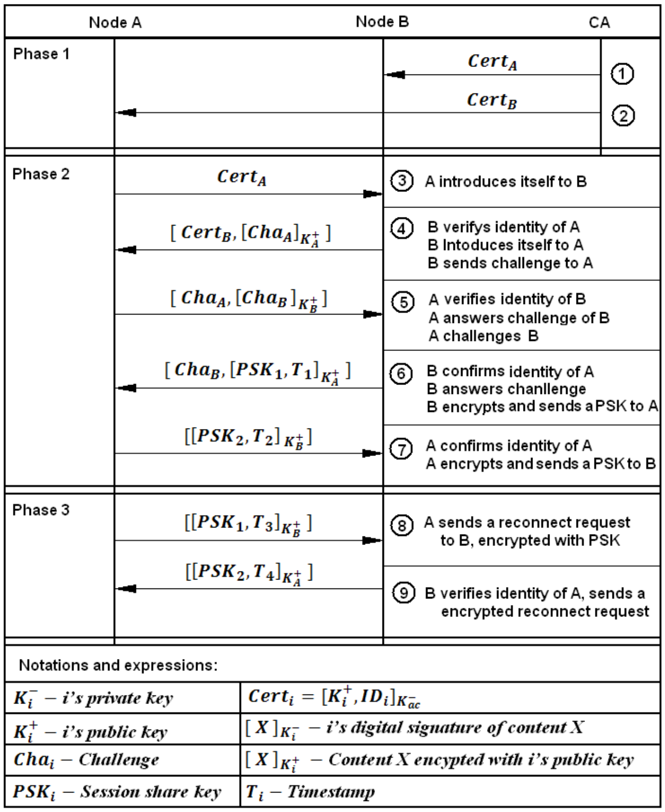

Node A
Node B
CA
Phase 1
$Cert_A$
1
$Cert_B$
2
Phase 2
$Cert_A$
3 A introduces itself to B
$[\, Cert_B, [Cha_A]_{K_A^+} \,]$
4 B verifys identity of A
B Intoduces itself to A
B sends challenge to A
$[\, Cha_A, [Cha_B]_{K_B^+} \,]$
5 A verifies identity of B
A answers challenge of B
A challenges B
$[\, Cha_B, [PSK_1, T_1]_{K_A^+} \,]$
6 B confirms identity of A
B answers chanllenge
B encrypts and sends a PSK to A
$[[PSK_2, T_2]\, _{K_B^+}]$
7 A confirms identity of A
A encrypts and sends a PSK to B
Phase 3
$[[PSK_1, T_3]_{K_B^+} \,]$
8 A sends a reconnect request
to B, encrypted with PSK
$[[PSK_2, T_4]_{K_A^+} \,]$
9 B verifies identity of A, sends a
encrypted reconnect request
Notations and expressions:
K_i^- – i's private key
$Cert_i = [K_i^+, ID_i]_{K_{ac}^-}$
K_i^+ – i's public key
$[\, X \,]_{K_i^-}$ – i's digital signature of content X
Cha_i – Challenge
$[\, X \,]_{K_i^+}$ – Content X encypted with i's public key
PSK_i – Session share key
T_i – Timestamp

Fig. 1. Node-to-node authentication

node A gives (the evidence)". By receiving this piece of information, a remote node is able to verify the routability of node A to node B, from time T_1 to T_1+*system defined valid time*.

The significance of the routability verification is, if all connections along the path from a node to its destination are verified, we can safely believe that the route is trustworthy. Each node cumulatively collects RCs and uses it to build a trusted routing map. Depending on the protocol used, a different distribution mechanism may be required. For proactive routing protocols such as OLSR, the identity certificate and RCs can be distributed along with periodic topology information advertisements, *i.e.* HELLO and TC messages. In reactive protocols such as AODV and DSR, the trusted routing information can be distributed in the route request and route reply messages. Each hop appends a relative RCs at the end of the control message or signs over the same piece of signature to reserve bandwidth. Figure 2 shows the cumulative routability verification mechanism in an example of a new node joining the network.

This module allows a remote node to built trustworthy routes without online assistance of trust authority or using return routability test mechanism [14]. The link status must be confirmed by both nodes hence compromised nodes cannot forge links that don't exist.

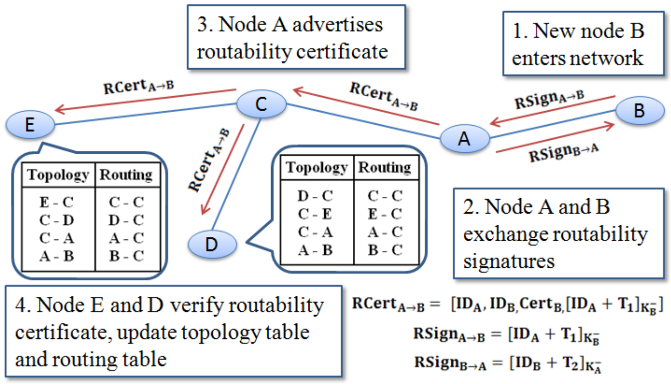

3. Node A advertises routability certificate
1. New node B enters network
$RCert_{A\to B}$
$RCert_{A\to B}$
$RSign_{A\to B}$
$RSign_{B\to A}$
$RCert_{A\to B}$
E
C
A
B
D
Topology
Routing
E - C
C - D
C - A
A - B
C - C
D - C
A - C
B - C
Topology
Routing
D - C
C - E
C - A
A - B
C - C
E - C
A - C
B - C
2. Node A and B exchange routability signatures
4. Node E and D verify routability certificate, update topology table and routing table
$RCert_{A\to B} = [ID_A, ID_B, Cert_B, [ID_A + T_1]_{K_B^-}]$
$RSign_{A\to B} = [ID_A + T_1]_{K_B^-}$
$RSign_{B\to A} = [ID_B + T_2]_{K_A^-}$

Fig. 2. Cumulative routability verification

III. Trusted Extension for OLSR

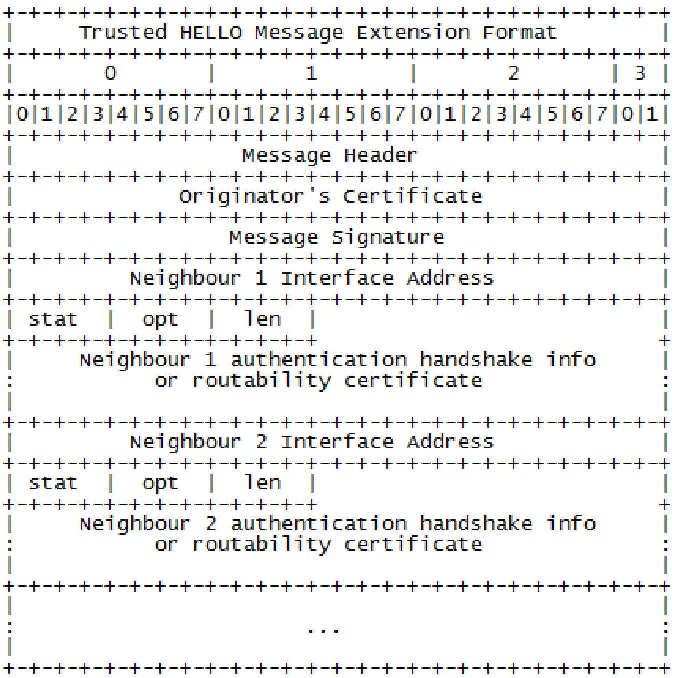
We applied the trusted routing framework described above to OLSR, a table-driven, proactive wireless routing protocol which has already been used as a solution for multihop VANET routing [15]. In this section the trusted OLSR extensions are explained in detail.

OLSR reduces the control traffic overhead by using Multipoint Relays (MPR). A MPR is a node's one-hop neighbour which has been chosen to forward packets. Instead of pure flooding of the network, packets are forwarded by a node's MPRs. This limits the network overhead, thus being more efficient than pure link state routing protocols. OLSR uses HELLO and Topology Control (TC) messages to discover and then disseminate link state information throughout the mobile ad-hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths [16]. We extended the HELLO and TC messages to carry additional information such as handshakes, certificates and digital signatures, to achieve message authentication, node-to-node authentication and routability verification.

A. HELLO Message Extension

The OLSR standard specifies three fuctions for the HELLO message: link sensing, neighbour detection and MPR selector set population [1]. One hop neighbour authentication is performed during the neighbour discovery phase using the node-to-node authentication mechanism discribed in section II-C. Two hop neighbours must be confirmed as reachable before calculating MPRs, hence the routability verification is performed after neighbour detection. With trusted extension, HELLO messages now served the following five purposes, in sequence:

1) Link sensing
2) Node-to-node authentication
3) Neighbour detection
4) Routability verification
5) MPR selector set population



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Trusted HELLO Message Extension Format        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |       1       |       2       | 3 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|0|1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Message Header                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Originator's Certificate               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Message Signature                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Neighbour 1 Interface Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| stat  |  opt  |  len  |                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+                         +
|     Neighbour 1 authentication handshake info     |
:           or routability certificate              :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Neighbour 2 Interface Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| stat  |  opt  |  len  |                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+                         +
|     Neighbour 2 authentication handshake info     |
:           or routability certificate              :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                   |
:                       ...                         :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 3. HELLO message extension format

The format of extended HELLO messages is shown in figure 3. Following the standard message header is a digital signature that guards the entire message. Different information may be appended after each neighbour interface address, depending on the authentication state that is indicated by the 4 bits *stat* field. The *opt* field indicates what contents are included; these can be handshake, identity certificate or RC. A *len* field is required to denote the size of the overhead since this is not constant.

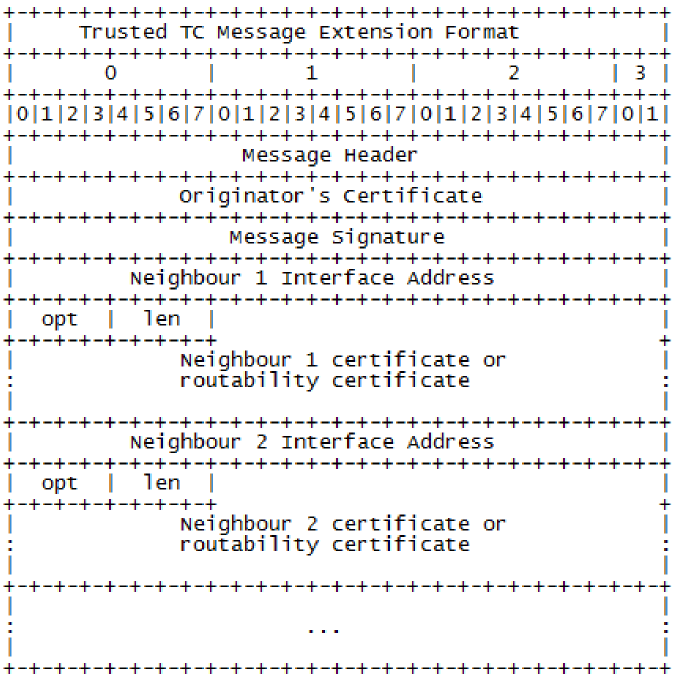
Assume node A wishes to authenticate with node B, the HELLO messages are created and handled as follow:

1) Node initiates the handshake by sending an empty HELLO message, its certificate is included if necessary.
2) Upon receiving a HELLO message from new neighbour node A, node B will include node A's interface address into the next HELLO message, followed by a challenge which is encrypted with node A's public key.
3) Node A answers the challenge in next HELLO message and a challenge to B will be sent with the same message.
4) Node B verifies the answer from node A; if it is correct, node A will be accepted as neighbour and added to the routing table, otherwise the conversation will be aborted.
5) Once two nodes are authenticated with each other, they exchange a RC as the proof of their relationship. The RC will be appended after the corresponding neighbour address in the HELLO message. A new RC will be re-issued before the previous one expires.
6) Other neighbours verify the link status of nodes A and B by checking their RCs. MPRs will be calculated after verification success.

B. TC Message Extension

In OLSR trusted routing extension, TC messages are used as carriers of node identity certificates and RCs. The TC message originator attaches corresponding RC to each neighbour in the advertised message. By verifying these extended TC messages, a node is able to build a trusted routing table of the network cumulatively and maintain an authentic topology map.

The format of TC message extension, as show in figure 4, is similar to HELLO messages but with different contents of the authentication field. The *opt* field indicates the presence of identity certificate and routability certificate. A *len* field is also required to indicate the size of overhead. Note the hop count and TTL fields are not included in the message signature because they will be changed by the relaying nodes.



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Trusted TC Message Extension Format           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |       1       |       2       | 3 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|0|1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Message Header                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Originator's Certificate               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Message Signature                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Neighbour 1 Interface Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  opt  |  len  |                                   |
+-+-+-+-+-+-+-+-+-+                                 +
|               Neighbour 1 certificate or          |
:               routability certificate             :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Neighbour 2 Interface Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  opt  |  len  |                                   |
+-+-+-+-+-+-+-+-+-+                                 +
|               Neighbour 2 certificate or          |
:               routability certificate             :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                   |
:                       ...                         :
|                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 4. TC message extension format

By receiving TC messages with RCs, a remote node performs the following steps to construct a trusted routing table:

1) Receiver authenticates the message by verifying the message signature. The message will be discarded if the authentication fails.
2) Check the validity of each RC, the neighbour who is confirmed to have an authentic link to the originator will be marked as ”pending” in the topology table. This is because the originator itself may not be

reachable. The neighbour with invalid RC will be discarded.

3) If the TC message originator address is found in the routing table, all verified neighbours in the TC message will be added to the routing table. Otherwise they must wait till the originator is reachable.
4) Each reachable address has a valid time in the topology map, which is determined by the RC. The valid time must be updated for every new RC received. If the valid time for an address is expired, the node is considered unreachable and will be removed from the routing table.

C. Certificate Distribution Scheme

Pre-distributing certificates of all members to each node is costly and impractical. In our scheme, certificates are distributed by the control messages at the time new nodes enter the network. Each node carries its own identity certificate which is issued by a CA before ITS entering the network. When two nodes first meet, they exchange the certificates to start node-to-node authentication handshake. If a remote node notices there are new nodes in the network, which may not have its certificate, the remote node includes the certificate in the next TC message. For those nodes who do not initiate TC messages, its MPR neighbour takes over this responsibility.

To further conserve bandwidth, RCs in HELLO and TC messages can be omitted. During the valid period of RCs, message originator does not need to include the same RCs in the control messages.

IV. Performance Evaluation of Trusted OLSR Extension

We evaluate the performance of Trusted OLSR in regards to resilience to attacks and overhead compared to the standard OLSR protocol.

A. Resilience to Attacks

As mentioned in earlier sections, our trusted routing framework has the functions of ensuring message integrity, entity trust and routability verification, in particular, the following attacks are addressed.

- **Illegal access:** A node must have a legitimate identity and be able to pass the node-to-node authentication handshake to access the network.
- **Impersonation attacks:** An attacker may perform a number of attacks , such as disturbing the network topology and attracting other node's data flow, by masquerading as a different node. Impersonation attack is also known as identity spoofing, *e.g.* spoofing the MAC or IP addresses. In our framework the message digital signature and node-to-node authentication components are used to prevent this type of attack.
- **Message modification:** As the TC messages are relayed by MPR selectors, a malicious relaying node may modify the passed-on messages and cause connectivity lost and conflicting routes. For example, the Advertised Neighbour Sequence Number (ANSN) attack mentioned in [17], where an attacker can modify the ANSN field of TC messages to a large value, so that any further message from the same originator will be dropped due to the loop free mechanism. A digital signature is attached to assure message integrity and hence solve such problems. The limitation for this scheme is that the variable fields are not protected. For instance attackers can change the TTL field of the TC message to a large value to limit its travel distance, or modify the hop count field to enduce other nodes to select/avoid a particular route.
- **Link spoofing:** In the OLSR protocol standard, each node builds its routing table relying on other nodes control messages, *i.e.* HELLO and TC messages. A compromised node can spoof the link relationship simply by including other nodes' IP addressses to its control message, even if it has no connection to these nodes. Routability verification thwarts such misbehaviour by requesting timestamped signatures from all neighbours, which requires a node to be in contact with advertised neighbours to acquire such information.
- **MPR selector isolation:** If a node is selected by its neighbour as MPR, this node is responsible for advertising MPR selector's information and forwarding message to this neighbour. However a malicious node may isolate its MPR selector neighbour by not including it in the control messages. Such violation can be regulated by a slight modification of RCs. After two nodes establish connection, they exchange an RC. Instead of signing link status of itself, the node signs for all neighbours of the other node, *i.e.* includes all IP addresses from the other node's HELLO message. Under this rule, a node must include all connected neighbours addresses in its control message to assure the RCs can be verified properly.

Note that our framework is not intended to prevent wormhole attacks, misbehaviour in relaying messages and other physical layer attacks. Some of the misbehaviours, such as intentionally dropping control messages, can be detected by other means, however network monitoring is beyond the scope of this paper.

B. Bandwidth and Computational Overhead

Depending on which public key scheme is chosen, the bandwidth and computational overheads that are added to the system may vary. Other factors that may affect the amount of overhead include handshakes, number of neighbours and whether the certificates are required. Overall, there are three sources of overhead: a message signature, certificates and routability certificates, which are included in selected HELLO and TC messages. The size of the certificate, $Lcert$, and routability certificate, L_{rc}, are given by equations 1 and 2.

$$L_{cert} = L_{ip} + L_{pub} + L_{time} + L_{sig} \quad (1)$$

$$L_{rc} = 2 * L_{ip} + L_{time} + L_{sig} \quad (2)$$

Where:

L_{sig} is the length of digital signature. 128-bit digital signatures provide sufficient security for short-live control messages.

L_{ip} is the length of IP address, which can be 32 bit for IPv4 or 128 bits for IPv6.

L_{pub} is the length of public key. In RSA standard, 1024-bit or 2048-bit public key are used while ECC uses 192, 233 or 521-bit keys [18].

L_{time} is the length of timestamp, assume 32 bit timestamp is used.

For the case when certificates and RCs are omitted, only the message signature is added to the original control message. If a node has n neighbours, n RCs and n certificates (if required) will be included in the control messages.

To evaluate computational complexity, we use results from [19] which provide a benchmark for various cryprographic schemes which may be used in our framework. The results are generated using the Crypto++® Library 5.6.0 which is used on an Intel Core 2 1.83 GHz processor. The most relevant results are shown in Table I.

Operation	Milliseconds/Operation
RSA 1024 Encryption / Decryption	0.08 / 1.46
DSA 1024 Signature /Verification	0.45 / 0.52
RSA 2048 Encryption / Decryption	0.16 / 6.08
RSA 2048 Signature / Verificatio	6.05 / 0.16
ECIES 233 Encryption / Decryption	21.17 / 12.15
ECDSA 233 Signature / Verification	10.62 / 12.80
MD5	0.0045 (per 1KB data)
SHA-1	0.0065 (per 1KB data)

TABLE I
Cryptographic Algorithm Benchmarks

We note that the computational cost of hash functions is converted to the time required to compute 1KB data, instead of using the amount of data the function can process per unit of time.

V. Conclusion and Future work

In this paper we have proposed a trusted routing framework which includes message authentication, node-to-node authentication and routability verification functionality. Digital signature of the control message ensures message integrity and originality; a secure neighbour discovery method is included in the node-to-node authentication module. The link status evidence mechanism included in the cumulative routability verification module regulates the behaviour of internal nodes. We have applied the proposed framework to the OLSR routing protocol, resulting in an OLSR extension which ensures trusted routing using only the routes witch include verified nodes. We have evaluated the resilience of the proposed extension to attacks and estimated the overhead compared to the standard OLSR protocol.

Within the routability verification module, there is currently a substantial amount of overhead added, which may result in scalability problems in large and dense network environments. In future work we plan to search for signature schemes that can reduce the overhead and computation time, *i.e.* to combine a number of neighbour signatures or batch signatures for faster verification. We also plan to apply this framework to different ad-hoc routing protocols such as AODV and DSR.

VI. ACKNOWLEDGEMENT

This research work has been supported by funding from National ICT Australia (NICTA).

References

[1] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF, RFC 3626, 2003.

[2] C. Perkins, E. Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF, RFC 3561, 2003.

[3] D. B. Johnson, D. A. Maltz, and Y. C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF MANET Working Group, RFC, February 2007.

[4] F. Janjua, S. Sultan, M. Muzaffar, Z. Ahmed, and S. Khan, "Authenticated Routing of Table Driven Protocol in an Ad-hoc Environment," in *INCC*, 2004.

[5] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network Magazine*, vol. 13, pp. 24–30, 1999.

[6] C.-P. Chang, J.-C. Lin, and F. Lai, "Trust-group-based authentication services for mobile ad hoc networks," in *ISWPS*, 2006.

[7] I. Doh, K. Chae, H. Kim, and K. Chung, "Security Enhancement Mechanism for Ad-Hoc OLSR Protocol," in *ICOIN*, 2006, pp. 317–326.

[8] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, and D. Raffo, "Securing the OLSR Protocol," in *IFIP*, 2003.

[9] M. Guerrero Zapata, "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing," Sep 2006.

[10] P. Papadimitratos and Z. J. Haas, "Secure Link State Routing for Mobile Ad Hoc Networks," 2003.

[11] Y.-C. Hu, A. Perrig, and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Wireless Networks*, no. 1-2, pp. 21–38, January 2005.

[12] D. Raffo, C. Adjih, T. Clausen, and P. Mehlethaler, "An Advanced Signature System for OLSR," in *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004, pp. 10–16.

[13] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," march 2005.

[14] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," June 2004.

[15] O. Mehani, R. Benenson, S. Lemaignan, and T. Ernst, "Networking Needs and Solutions at Imara," in *ITST*, 2007.

[16] K. Holter, "Comparing AODV and OLSR," 2005, http://folk.uio.no/kenneho/studies/essay/essay.html.

[17] D. Raffo, "Security schemes for the OLSR protocol for ad hoc networks," Ph.D. dissertation, Université Paris 6 and INRIA Rocquencourt, 2005.

[18] S. Shen and M. Vanderveen, "ECC Support for SEND/CGA," IETF, Tech. Rep., 2009, internet-Draft.

[19] W. Dai, "Speed Comparison of Popular Crypto Algorithms," 2009, crypto++® Library 5.6.0 Benchmarks.