Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2413–2427
computer communications
www.elsevier.com/locate/comcom

Trust-based security for wireless ad hoc and sensor networks ☆

A. Boukerch [a,*], L. Xu [a], K. EL-Khatib [b]

[a] PARADISE Research Laboratory Site, University of Ottawa, Ottawa, Canada
[b] University of Ontario, Institute of Technology, 2000 Simcoe Street, North Oshawa, Ont., Canada L1H-7K4

Available online 6 May 2007

Abstract

Wireless sensors networks are susceptible to a large number of security threats, and because of the communication, computation and delay constraints of most applications that run on top of these networks, traditional security mechanisms cannot be used. Trust and reputation have been recently suggested as an effective security mechanism for open environments such as the Internet, and considerable research has been done on modeling and managing trust and reputation. Using the trust and reputation management scheme to secure wireless sensor networks (WSNs) requires paying close attention to the incurred bandwidth and delay overhead, which have thus far been overlooked by most research work. In this paper, we propose a novel agent-based trust and reputation management scheme (ATRM) for wireless sensor networks. The objective of the scheme is to manage trust and reputation locally with minimal overhead in terms of extra messages and time delay. Throughout the entirety of this paper, we describe our scheme and prove its correctness. We will also present our extensive performance evaluation results, which clearly show that trust and reputation can be computed in wireless sensor networks with minimal overhead.

Keywords: Wireless sensor networks; Mobile agent; Trust; Reputation; Overhead; Delay

1. Introduction

Wireless sensors are small and cheap devices powered by low-energy batteries, equipped with radio transceivers, and responsible for responding to physical stimuli, such as pressure, magnetism and motion, by producing radio signals. They are featured with resource (e.g., power, storage, and computation capacity) constraints and low transmission rates. Wireless sensor networks (WSNs) are collections of such wireless sensors that are deployed (e.g., using aircraft) in strategic areas to gather data about the changes in their surroundings, to report these changes to a data-processing center (which is also called a data sink), and possibly to respond to these changes. The processing center can be a specialized device or just one of the sensors, and its function is to analyze the collected data to determine the characteristics of the environment or to detect events. Mass-produced intelligent sensors and pervasive networking technology enable WSNs to be widely applied to various applications, ranging from military to civilian fields; examples of these applications include military surveillance, target tracking, traffic monitoring, and building safety monitoring, to list a few.

Though wireless sensor network holds promise for a large number of applications, the problem of securing these networks has been a roadblock to their large scale adoption and deployment, and the research field of securing wireless sensor networks is still in its infancy. While wireless sensor networks are prone to the security threats of conventional networks, these networks are subject to additional threats; these additional threats result from the sensor nodes' intrinsic characteristics, mainly the limited communication bandwidth, computation power, memory and battery capacity, and deployment environment. Traditional safety mechanisms for providing confidentiality, authentication, and availability are not efficient in wireless sensor networks where network sensor nodes have limited communication bandwidth, CPU cycles, memory, and battery capacity.

☆ This work was partially supported by Canada Research Chair Program, NSERC, CFI, and Ontario Distinguished Researcher Awards.
* Corresponding author. Tel.: +1 613 301 9476.
E-mail address: boukerch@site.uottawa.ca (A. Boukerch).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.022

These traditional safety mechanisms come at the cost of computation complexity of encryption algorithms, memory usage for storing security information, and network bandwidth for key synchronization and certificate distribution and revocation. Moreover, wireless sensors networks are deployed in open, unattended, and physically insecure environments where an adversary can easily capture nodes and subsequently use these nodes to attack the whole network. Building tamper-proof sensor nodes is not a viable solution since these nodes are deployed in large numbers, and adding to the building cost of these nodes is not a welcome solution. Finally, the radio communication between network sensor nodes makes these networks susceptible to all possible attacks on wireless communication environments.

Trust and reputation have been recently suggested as an effective security mechanism for open environments such as the Internet, and considerable research has been done on modeling and managing trust and reputation. Some research work has shown that rating nodes' trust and reputation is an effective approach in distributed environments to improve security [2,13,27,33], to support decision-making [3,23,43], and to promote node collaboration [10,19,29,32]. Using the trust and reputation management scheme to secure wireless sensor networks (WSNs) requires paying close attention to the incurred bandwidth and delay overhead, which so far have been overlooked by most research work. Searching nodes' reputation in a network with a central authority, or even in cases where the reputation information is distributed in a network with plentiful communication resources, is not difficult. The absence of any centralized authority in wireless sensor networks and the bandwidth limitation of these networks make it challenging to trace nodes' reputation accurately. Flooding the network with request messages is a useful tool for data-searching in a fully distributed environment. However, since message transfer consumes both bandwidth and energy, trust and reputation management schemes that generate large amounts of traffic by flooding the network with request messages are not desirable in wireless sensor networks since these are known for their bandwidth and energy constraints. In addition, because trust and reputation information is usually requested by nodes before they start communicating with each other, trust and reputation management schemes with poor trust and reputation acquisition latency are not acceptable in situations with strict real-time requirements, like battlefields and emergency rescue. Under these circumstances, we propose a novel agent-based trust and reputation management scheme (ATRM) for wireless sensor networks; the main objective of our scheme is to manage trust and reputation with minimal overhead in terms of extra messages and time delay. The ATRM is based on a clustered WSN with backbone and on a mobile agent system; it introduces a trust and reputation local management strategy with help from the mobile agents running on each node. That is, a nodes trust and reputation information is stored on the node itself and managed by the local mobile agent of the node. The benefit of a local management scheme for trust and reputation is that there is no need for centralized repositories, and the nodes themselves are able to provide their own reputation information whenever requested; therefore, reputation computation and propagation is done without network-wide flooding and with no acquisition-latency.

The rest of this paper is organized as follows: Section 2 reviews some related work on trust and reputation management systems. Section 3 briefly describes hierarchical wireless networks and mobile agent systems. Section 4 gives a detailed presentation of the ATRM. In Section 5, we prove the correctness of the ATRM, and in Section 6 we study the complexity of the ATRM. Section 7 shows the results of our performance evaluation experiments on the ATRM, and our conclusion is drawn in Section 8.

2. Previous and related work on securing systems using trust and reputation

Trust, as an integrative component of human society, is an abstract matter that we deal with in our everyday lives, but its relevant research is quite narrow, and therefore causes a lack of coherence in its definitions between different disciplines [28]. In computer science, there are many definitions and models for trust, such as the ones given in [1,2,14]. Based on the previous work described in [1,2,14], the notion of trust to be used throughout this paper is briefly defined as: *trust is the degree of belief about the future behavior of other entities, which is based on the ones the past experience with and observation of the others actions*. The properties of trust are summarized as: subjectivity [30,35], non-transitivity [12], temporalness, [4], contextualness and dynamicity, as well as non-monotonicity [2]. Additionally, we can derive from subjectiveness and non-transitiveness another characteristic of trust, that it is unidirectional.

Reputation is another complex notion that spans across multiple disciplines. It is quite different from but easily confused with trust. Although in some research papers, e.g. [2,9,16] to mention just a few, authors attempt to define and model reputation in different ways, most of them share a common agreement that the basis of an entitys reputation is the trust the others hold in that entity. Likewise, the reputation in the ATRM is defined as: *in a community, the reputation of an entity is the global perception of the entitys behavior norms based on the trust that other entities hold in the entity.* Though some previous work [26,31] has pointed out that reputation should be subjective, we believe it is reasonable to consider that reputation is objectivity [17] since ones reputation is derived from *all the others'* opinions. Other characteristics of reputation include temporalness [26], contextualness [16,31], and dynamicity, as well as non-monotonicity.

Early trust-based security systems dealt only with certain aspects of trust management; for example, PKI X.509 [21] and PGP [45] are designed to solve the problem

of finding a suitably trustworthy copy of the public key of someone. But it was Blaze, Feigenbaum, and Lacy [8] who were the first to formally introduce "trust management" as a separate component of security in network services and gave an overall definition of the trust management problem. The authors proposed a unified decentralized trust management system, Policymaker, which was based on a simple language for describing security policies, credentials and relationships. Policymaker works like a database engine. Its basic function is to process queries, each of which is a request to determine if a particular public key or a set of particular public keys are permitted to perform a particular action according to local policy. A Policymaker query has the following form: key_1,key_2, ..., key_n REQUESTS *ActionString*, where *ActionString* describes a proposed trusted action and is application-specific. Depending on processing results, Policymaker may accept/reject the requested actions or return additional suggestion to make the requested actions acceptable. Similar systems include Keynote [7], RT [24], and TPL [20].

Gupta, Judge, and Ammar [17] proposed a partially distributed reputation system for P2P networks. This system involves a central reputation computation agent (RCA) that is presumably trusted by every other node. In order to secure reputation computation and to prevent the nodes' modification of their locally-stored reputation information, public-key cryptography is utilized. The RCAs public key is required to be available to every node in the network, and each node generates a (Public-Key, Private-Key) pair and registers it with the RCA. The digest of the public key of a node is used by the RCA as the nodes identification. A node's reputation computed by the RCA is encrypted with RCAs private key and locally stored by the node itself. The RCA does not participate in nodes' transaction process, but it is periodically requested by nodes for credits for their contribution in the system based on two proposed reputation computation schemes, Debit–Credit (DCRC) and Credit-Only (CORC).

Kamvar, Schlosser, and Garcia-Molina [23] proposed a reputation management algorithm, called EigenTrust, for P2P networks. In EigenTrust, there is a distinction between a nodes local trust and its global reputation. For any two nodes n_i and n_j, the local trust n_i holds in n_j and is normalized as $\frac{\max(s_{ij},0)}{\sum_j \max(s_{ij},0)}$ where s_{ij} represents the local trust rating depending on the Quality of Service (QoS) n_j provided. The reputation of n_j is based on the local trust values, which are assigned to n_j by other nodes (e.g., node n_i) and weighted according to the trust of the assigning nodes. Trust aggregation is done using a transitive trust mechanism: a *requester* asks its friends for their opinions about the evaluated node, and then it asks for the opinions of its friends' friends, and then it asks for the opinions of its friends' friends' friends, and so on. This kind of asking process ends after a pre-determined number of iterations.

Jurca and Faltings [22] proposed an incentive-compatible reputation mechanism, which introduces a side-payment scheme and cryptographically protects the integrity of reputation information. Side payments are organized through a set of always-existing broker agents, named *R*-agents, which sell and buy reputation information from agents. The payoff for an agent selling reputation information to an *R*-agent depends on whether the provided information is consistent with future reports on the same agent. Two agents that are about to trade use a certain algorithm to select an *R*-agent, ask the selected *R*-agent for the reputation of their partners, and pay for the reputation information. An agent that loses all its reputation money cannot use the reputation service any more. Having known the reputation of their partners, agents will decide if they should have the transaction with their partners. After the transaction is over, agents get transaction payoff from which they can exactly determine the behavior of their partners and submit a report to the selected *R*-agent. Based on the outcome of the transaction, agents also update their opinions about the effectiveness of different *R*-agents.

Taking into account the dynamicity of MANETs, Ren and his colleagues [35] proposed a certificate-based trust establishment scheme. They addressed the trust establishment with dynamically joining and leaving nodes under the assumption of existing sparse social relationships among nodes. The scheme requires a bootstrapping phase during which a secret dealer is required. In the first part of the bootstrapping phase, every member node obtains its secret short list from the secret dealer, which includes k bindings of a node identifier and its corresponding public key. The value of k is carefully chosen depending on the group size and may vary slightly from node to node. The second part of the bootstrapping phase is a certificate issuance process, where all member nodes generate certificates for the received bindings from their own domain and store the certificates locally. After the bootstrapping phase, the network becomes fully functional with no need for the secure dealer, and the trust establishment for any two nodes turns into a certificate chain detection problem in a certificate graph.

Theodorakopoulos and Barasa [42] proposed a trust evaluation mechanism for MANETs without the need for a centralized infrastructure (e.g., the secure dealer in [35]). In their mechanism, trust evaluation is viewed as a generalized shortest path problem on a weighted directed graph where vertices represent nodes and weighted edges correspond to the *opinions* that one end node has about the other end node in the edge direction. *Opinions* are assigned to edges by nodes based on their local observation and on their own criteria. Every *opinion* consists of two values, trust value and confidence value. The former is a nodes trust estimation while the latter reflects the accuracy of the trust value assignment. In such a graph, an indirect trust relation without previous immediate experience is established by the theory of semirings.

Buchegger and Boudec [9] presented a reputation system for MANETs and P2P networks, which is featured with the elimination of the affection on reputation from incompatible recommendation. In their system, each node n_i

maintains three types of data about everybody else, e.g. node n_j, that it cares about. These three types of data are the trust rating T_{ij}, the reputation rating R_{ij}, and the first-hand observation summary F_{ij}. T_{ij} represents n_is confidence in how possible it is that the first-hand observation provided by n_j is true. R_{ij} reflects n_i's opinion about how trustworthy n_j is in the transactions. F_{ij} is a summary record of n_is immediate observations about n_j. When n_i makes first-hand observation about n_js behavior, R_{ij} and F_{ij} are updated accordingly. And, n_i periodically publishes F_{ij} as its recommendation to n_j. When n_i receives another node n_ks first-hand information about n_j, F_{kj}, it may or may not update R_{ij} based on T_{ik} (i.e., if n_k is trustworthy enough) and R_{ij} (i.e., how different R_{ij} and F_{kj} are) by a modified Bayesian approach and based on a linear model merging heuristic. According to the decision, T_{ik} is also updated. In this algorithm, only first-hand information is published, and in order to reduce traffic, the publication is confined to a subset of nodes.

Liu and Issarny [26] introduced another reputation mechanism for MANETs, where service reputation and recommendation reputation is differentiated. By using both recommendation and reputation, this mechanism is able to distinguish between truth-telling and lying nodes and thus to be resilient against attacks of defamation and collusion. Reputation is managed distributively by an experience manager, a recommendation manager, and a reputation manager running on each node. The experience manager takes the responsibility to store the previous experiences with other nodes. The recommendation manager is in charge of storing other nodes' recommendations, exchanging reputation information with other nodes, and managing a table of recommendation reputation of recommenders. The reputation manager computes and manages the service reputation of a node, referring to the inputs from both the experience manager and the recommendation manager. Periodically, the recommendation manager contacts other nodes for a reputation information exchange. If a recommendation manager receives a recommendation exchange request from another node, it may accept or reject the request depending on whether the requesters recommendation and reputation are above a pre-defined threshold value.

3. Hierarchical wireless networks and mobile agent systems

In this section, we are going to present some background knowledge about hierarchical sensor wireless networks and mobile agent systems. Both of these technologies are important for our scheme.

3.1. Hierarchical wireless networks

For a wireless network with n nodes capable of transmitting at Wbits/sec, according to [18], the throughput, T, for each node under optimal conditions is

$$T = \Theta\left(\frac{W}{\sqrt{n}}\right) \tag{1}$$

According to the above formula, the number of nodes should be kept as small as possible in order to gain efficient throughput performance. But in WSNs, there are usually thousands of nodes, and scalability is therefore an important issue.

Clustering is an effective approach to improving network scalability and throughput, and it has been studied by a number of researchers [5,6,15,25]. Using clustering algorithms, nodes are grouped into clusters, and within each cluster a node is elected as a cluster head. Cluster heads together form a higher-level network, upon which clustering can again be applied. After a number of recursive iterations, a clustering algorithm constructs a multi-level network structure. This structure facilitates communication and makes it possible to restrict bandwidth-consuming network operations like flooding only to the intended clusters. Restricting data communication to clusters can lead to reduced network traffic and hence improved network scalability. When cluster heads have the same communication capability as regular nodes, communication between cluster heads has to be done through a sequence of intermediate nodes, and thus the improvement in scalability can be limited. This problem has led to the proposal of mobile backbone networks, which will be introduced in the following paragraphs.

The main difference between traditional clustered wireless networks and mobile backbone networks is the application of radios of different ranges. In mobile backbone networks [44], cluster heads are required to be predeployed backbone nodes, which are capable of communicating through radios of different ranges and are connected directly through long-range radios instead of by message relay. The nodes in the same level share the same communication channel, but radios in different levels use different frequencies and channel resources. Nodes communicate only with other nodes belonging to the same cluster, and inter-cluster communication has to be done through cluster heads in the higher-level backbone network. Using this approach, each cluster can be considered an independent small-sized wireless network, and the higher-level backbone network is another wireless network linking the clusters together.

Assume that there is a two-level mobile backbone network where the number of nodes is n and the number of clusters is m. Furthermore, denote by W_1 the transmission rate in the lower-level network and by W_2 the transmission rate in the upper-level network. Then, the number of nodes in each cluster is n/m on average, and according to Formula 1, the per node throughput of clusters (in the lower-level) $T_1 = \Theta(\frac{W_1}{\sqrt{n/m}})$, and the per node throughput of the backbone network (in the upper-level) $T_2 = \Theta(\frac{W_2}{\sqrt{m}})$. When n is fixed, it is clear that T_1 increases with the growth of m while T_2 decreases, and thus it is necessary to determine

the optimal m which makes the mobile backbone network achieve optimal overall throughput. Since cluster heads are required to be backbone nodes, the optimal m implies the optimal number, referred to as M, of backbone nodes needed. According to [44], M is computed as:

$$M = \frac{W_2}{W_1}\sqrt{n} \tag{2}$$

Since this result is achieved under optimal conditions, the number of backbone nodes is actually less than it is in the real world. A backbone network construction algorithm, RCC, and its performance evaluation is presented in [44].

3.2. Mobile agent systems

Mobile agents are the programs launched by network users to accomplish certain given tasks while migrating from one computation environment (a computer) to another. They can be widely employed in many fields like electronic commerce, secure brokering, distributed information retrieval, and telecommunication network services. Mobile agents are featured with autonomy, asynchrony, adaptivity and communicability. After a mobile agent runs on a computer, it accesses the local resources, interacts with the local execution environment, senses the dynamic changes in the network and acts accordingly, following its self-contained rules. If necessary, it can also communicate with other mobile agents by means of messages to collaboratively achieve a common goal. Since the required data is locally accessed and the computation logic is encapsulated inside the mobile agents, the transfer of control and data messages is minimized, and thus the possibility of network congestion decreases. Using mobile agents to access distributed data creates an efficient method for data distribution, aggregation, and sharing in distributed network environments with bandwidth constraints.

So far, a large number of research activities on mobile agents have been carried out, and many mobile agent systems, such as Grasshopper, Anima, Odyssey and Planet, have already been developed by different institutes and companies. However, security and fault-tolerance problems in mobile agent systems always remain, since mobile agents are designed to run on remote computers and to travel over large-scale networks. On the one hand, mobile agents' computation logic and accumulated data are both at risk from attacks from their host computers, while, on the other hand, mobile agents roaming around in the network can be used by their hosts through the embedding of malicious codes to assault other machines. Therefore, mobile agent systems should be secured against unauthorized analysis and modification. Additionally, since both node and communication failure are common phenomena, mobile agents are very likely to be lost during their trips. Due to software faults in mobile agents or in their execution environment, mobile agents may crash or have an incorrect statuses. In these cases, mobile agent systems should be able to detect the loss and failure of mobile agents, to recover mobile agents from incorrect status, and to ensure mobile agents' arrival at their destination. Although many security mechanisms [37,39–41] and fault-tolerance models [11,34,36,38] have been proposed and developed for mobile agents systems, there are still a number of issues that need to be addressed before having a robust mobile agent system.

4. Agent-based trust and reputation management scheme

Existing trust management schemes like [8,17,22,23,45] were designed originally for wired networks (e.g., P2P), and are thus not suitable for wireless sensor networks where nodes have resource constraints. The schemes in [9,26,35,42] are developed for wireless networks, but they focus mostly on trust and reputation modeling and seldom emphasize on the network performance problem. The most commonly used approaches by these schemes for acquiring reputation can be classified into two groups; on-demand and periodical. As trust information is distributed into the entire network, reputation computation requires network-wide flooding for trust aggregation. Whether reputation computation is done periodically or on-demand, it will incur bandwidth usage and energy consumption. Although the scheme presented in [9] restricts flooding to a subset of nodes, how to determine the subset such that it covers all the nodes (or a sufficient number of nodes) holding the required trust information becomes a problem.

In this section, we propose a novel agent-based trust and reputation management scheme (ATRM) for WSNs. The main objective of the ATRM is to effectively manage trust and reputation with minimal overhead in terms of extra messages and time delay. The notations to be used for describing the ATRM can be found in Table 1.

4.1. Overview

The ATRM is based on a clustered WSN with backbone, and its core is a mobile agent system. Differing from traditional trust and reputation management systems, ATRM requires that a node's trust and reputation information be stored respectively in the forms of *t*-instrument and *r*-certificate by the node itself. Obviously, nodes cannot manage and compute their own trust and reputation. So, ATRM further requires that every node locally hold a mobile agent that is in charge of administrating the trust and reputation of its hosting node. In this sense, mobile agents provide nodes a "one-to-one" trust and reputation management service.

In ATRM, an arbitrary transaction is defined as the process of interaction between two nodes, the *requester* and the *provider*, and it is triggered by the *requester* and may be accepted/rejected by the *provider*. Before starting any transaction, the *requester* asks its local mobile agent to obtain the *r*-certificate of the *provider* by directly querying the *provider*'s local mobile agent. Based on the provider's *r*-certificate, the *requester* decides whether or not to start the

Table 1
Notations for ATRM

Name	Description
n_i	An arbitrary node in the original network
n'_j	An arbitrary node in the backbone network
$C_\star$	An arbitrary context
ID(n_i)	The ID of node n_i
TRA	A trust and reputation assessor agent
AK	The symmetric key held by a TRA
AL	The agent launcher
t-Instrument	A trust instrument
r-Certificate	A reputation certificate
TRAn$_\star$	An arbitrary transaction
n_{req}	The *requester* of transaction Tran$_\star$
n_{pro}	The *provider* of transaction Tran$_\star$
Tab$_{\text{eval}}$	Trust evaluation table
Tab$_{\text{instr}}$	*t*-Instrument table
Buf$_{\text{cert}}$	*r*-Certificate buffer
CNTER	Received Lowversion message counter
$t_{i,j}$	The trust evaluation made by n_i on n_j
$H(M)$	The hash value of the message M
$E_{\text{AK}}(M)$	The Message M is encrypted with AK
TI(n_i,n_j,$C_\star$)	The *t*-instrument issued by n_i to n_j for $C_\star$
RC(n_i)	The *r*-certificate of node n_i
δ_{cert}	*r*-Certificate acquisition timeout
δ_{instr}	*t*-Instrument issuance timeout
δ_{valid}	The lifetime of a trust evaluation
θ_{cnt}	The threshold value of *CNTER*
θ_{ACK}	The maximum number of acknowledgment retries
θ_{cert}	The maximum number of *r*-certificate acquisition retries
θ_{instr}	The maximum number of *t*-instrument issuance retries

transaction. After a transaction is finished, the *requester* makes a trust evaluation on the *provider* based on the quality of service it gets from the *provider* during the transaction, and then submits the evaluation to its local mobile agent which then accordingly generates a *t*-instrument for the *provider* and sends the *t*-instrument to the *provider*'s local mobile agent. Based on the collected *t*-instruments, a mobile agent periodically issues its hosting node updated *r*-certificates.

4.2. Network model

In this paper, we use a network model based on sufficient backbone nodes with multiple long-range radios that are randomly scattered in the WSN. Every node has a unique ID by which it can be distinguished from others. Through a secure routing protocol, all the nodes (including backbone nodes) together compose a low-level network using a short-range radio, which is referred to as the *original network*. Likewise, the backbone nodes additionally constitute a high-level network, *the backbone network*, via a long-range radio. The original network is dynamically partitioned into a number of clusters by an effective clustering algorithm running on each node, each cluster has a backbone node elected as cluster head. Fig. 1 shows an example of such 2-level WSNs. Both backbone and non-backbone nodes may fail or become unavailable due to a system crash, power exhaust, or any other reason; however, the rest of the original network and backbone network should still be connected.

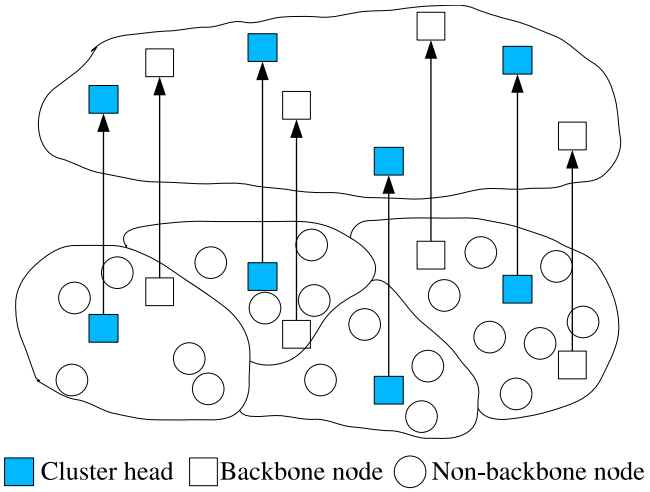

Fig. 1. A clustered WSN with backbone.

4.3. Assumptions

Since mobile agents are designed to travel over the entire network and run on remote nodes, they must be launched by trusted entities. And clearly, compromised mobile agents may not provide the expected services and can actually constitute a threat to network security. Therefore, in ATRM, we assume (1) that there is a trusted authority that is responsible for generating and launching mobile agents, and (2) that mobile agents are resilient against the unauthorized analysis and modification of their computation logic.

4.4. System architecture

The architecture of ATRM consists of four key components: agent launcher (AL), trust and reputation assessors (TRAs), trust instruments (*t*-instruments), and reputation certificates (*r*-certificates).

4.4.1. Agent launcher (AL)

An AL is an authority responsible for generating and launching TRAs into the network. It may be a piece of software, a node, or an organization, and it could be either inside or outside the network. In ATRM, we assume that there is only one AL that is always-existing and trusted by every node. The AL launches one TRA each time in a broadcast fashion into the backbone network. Before launching a TRA, the AL associates it with a symmetric secret key, AK, and a monotonically increased version number (starting from 1). The purpose of AK is to secure trust aggregation and reputation propagation, while the version number is used to support agent consistency verification. In case the AK of the current TRA is stolen or broken, the AL may periodically launch a new TRA with a

higher version number and a fresh AK to replace old ones according to application-specific security requirements.

4.4.2. Trust and reputation assessor (TRA)

A TRA is a mobile agent generated by the AL. It is designed to be distributed into every node and to provide its hosting node with a trust and reputation management service. Each node will hold a replica of the TRA's current version. For an arbitrary node n_i, its replica TRA, TRA(n_i), locally maintains four data structures, i.e. a trust evaluation table Tab_{eval}, a *t*-instrument table $\text{Tab}_{\text{instr}}$, a *r*-certificate buffer Buf_{cert}, and a LowVersion message counter *CNTER*. The trust evaluations that n_i recently made on other nodes are kept in Tab_{val}, while the *t*-instruments issued to n_i by the local replica TRAs of other nodes are stored in $\text{Tab}_{\text{instr}}$. Buf_{cert} accommodates n_i's *r*-certificate last issued by TRA(n_i). As for *CNTER*, it is incremented whenever TRA(n_i) receives a LowVersion message from a node for the first time since the last *CNTER* resetting. The relation between a node and its local TRA is shown in Fig. 2, where the dashed lines with double arrows represent the interaction between the TRA and its host.

As illustrated in Table 2, Tab_{eval} is composed of four fields, ID, CTX, EVAL, and TS, among which ID and CTX together constitute the primary key of the table. Field ID contains the IDs of the evaluated nodes; field CTX implies trust contexts; and field EVAL stores the trust evaluation values; field TS holds the time when evaluations are made. For any node n_j, its entry for context $C_\star$ in the Tab_{eval} of node n_i is denoted by $\text{Entry}_{\text{eval}}(n_j, C_\star)$.

Table 3 shows the structure of the $\text{Tab}_{\text{instr}}$ of n_i. $\text{Tab}_{\text{instr}}$ contains five fields, including ID, CTX, INSTR, TS and ACK, of which ID and CTX together constitute the primary key of the table. Field ID contains the IDs of *t*-instrument *issuers*; field CTX implies trust contexts; and field INSTR stores *t*-instruments. Field TS holds the time when *t*-instruments are issued, while field ACK reflects how may

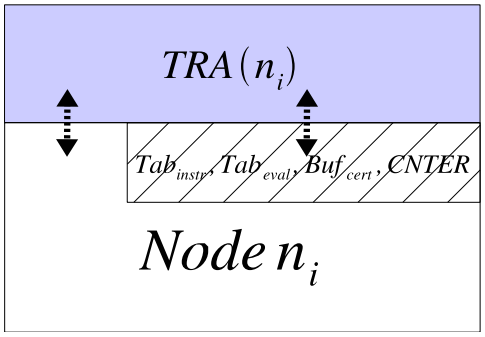

Fig. 2. A node and its local replica TRA.

Table 2
The Tab_{eval} of node n_i

ID*	CTX*	EVAL	TS
ID(n_j)	$C_\star$	t_{ij}	T

Table 3
The $\text{Tab}_{\text{instr}}$ of node n_i

ID*	CTX*	INSTR	TS	ACK
ID(n_j)	$C_\star$	TI($n_j, n_i, C_\star$)	T	1

times a *t*-instrument issuance is acknowledged, and its default value is 1. For any node n_j, its entry for context $C_\star$ in the $\text{Tab}_{\text{instr}}$ of node n_i is denoted by $\text{Entry}_{\text{instr}}(n_j, C_\star)$.

A replica TRA stays on its host until it is replaced by the replica of a higher-version TRA, and in the meantime it offers its host the trust and reputation management service. When TRA replacement takes place, the new local TRA will take over all the data structures maintained by the old one and reset *CNTER* to 0. A detailed description of the TRA's functionality can be found in Section 4.5.2.

4.4.3. Trust instruments (t-instruments)

A *t*-instrument is a segment of data that is organized with a special structure and issued by the local replica TRA of a node (*issuer*) to another node (*issuee*). It is stored in the $\text{Tab}_{\text{instr}}$ on its *issuee* node. Considering any two nodes n_i and n_j, the *t*-instrument issued by TRA(n_i) to n_j under context $C_\star$ is defined as:

$$\text{TI}(n_i, n_j, C_\star) = E_{\text{AK}}(D, H(D)) \tag{3}$$

where $D = (\text{ID}(n_i), \text{ID}(n_j), C_\star, T, t_{i,j})$ and T is a time-stamp implying the time when the *t*-instrument is issued. From the above definition, we can see that a *t*-instrument implicitly indicates the temporalness property of trust by the use of a time-stamp T. *t*-instrument issuance is driven by transactions, and it involves message transmission between the local replica TRAs of the *issuers* and *issuees*. A more detailed description of *t*-instrument issuance can be found in Section 4.5.2.

4.4.4. Reputation certificates (r-certificates)

A *r*-Certificate is a segment of data that is organized with a special structure and issued by a replica TRA to its host. A node's *r*-certificate is locally stored in the Buf_{cert} on the node and periodically refreshed by the local replica TRA. If there are k concerned contexts, for any node n_i, its *r*-certificate is defined as

$$\text{RC}(n_i) = E_{\text{AK}}(R, H(R)) \tag{4}$$

where $R = (\text{ID}(n_i),\ T,\ ((r_1, C_1),\ (r_2, C_2), \cdots,\ (r_k, C_k)))$ and T is a time-stamp implying the time when the *r*-certificate is issued. This formula is explained as follows: n_i's reputation is r_1 under context C_1, r_2 under the context C_2, $\cdots$, and r_k under context C_k at time point T. This definition implicitly reflects the contextual and temporal properties of reputation. Descriptions of *r*-certificate acquisition and issuance can be found in Section 4.5.2.

4.5. Phases of the ATRM

The execution of ATRM involves two phases; *network initialization* phase and the *service-offering* phase. As soon

as ATRM starts, the *network initialization* phase is initiated. The purpose of this phase is to distribute a TRA to every node. What follows is the *service-offering* phase during which the trust and reputation service is provided. In this subsection, we will go through the details of these two phases.

4.5.1. Network initialization phase

The *network initialization* phase consists of two stages. In the first stage, the AL launches a TRA in the backbone network in a broadcast fashion. Considering an arbitrary node n'_j in the backbone network, when it receives a TRA for the first time, n'_j makes a replication of the TRA and then forwards the TRA to all its immediate neighbors in the backbone network. If n'_j receives an already-received TRA, it just discards the TRA. Once n'_j has a replica TRA, it enters the second stage. In the second stage, n'_j checks wether it is a cluster head itself. If so, n'_j broadcasts its replica TRA within its cluster in order to distribute the replica TRA to all its cluster members, otherwise it keeps silent. The *network initialization* phase is run at the beginning of the execution of the ATRM, and it may also be re-run later from time to time, to update the replica TRAs depending on application-specific security requirements.

4.5.2. Service-offering phase

As long as a node has a local replica TRA, the trust and reputation service provided by the replica TRA is available to the node, and thus we say that the node is in the service-offering phase. The trust and reputation service is composed mainly of four types of sub-services: *r*-certificate acquisition, *t*-instrument issuance, *r*-certificate issuance, and trust management routine. The first two sub-services are transaction-driven and involve the message transmission between the transaction *requester* and *provider*, whereas the other two sub-services involve merely the local processing periodically performed by the replica TRA of each node.

Because of the asynchronous execution, nodes are unlikely to enter the *service-offering* phase at the same time. Specifically, it is possible that some nodes do not yet have a local replica TRA or have an inconsistent version of the TRA when they are asked by other nodes for *r*-certificates. The inconsistency of two replica TRAs means that the two replica TRAs derive from different versions of the TRA and thus have distinct secret keys. Inconsistent replica TRAs are not able to correctly process the *t*-instruments and *r*-certificates issued by each other. Clearly, the asynchronous execution may lead to the failure of the trust and reputation management service. In the following paragraphs, we are going to explain how the four types of sub-services are delivered in spite of the presence of failures.

4.5.2.1. r-Certificate acquisition. *r*-Certificate acquisition is initiated by TRA(n_{req}) in the pre-transaction process just before the official start of $\text{Tran}_\star$. Its objective is to obtain the reputation of n_{pro}, based on which n_{req} is able to decide whether to really start $\text{Tran}_\star$ with n_{pro}. Because the *r*-certificate reflecting n_{pro}'s reputation is locally stored by n_{pro} and managed by TRA (n_{pro}), n_{req} just commissions its own replica TRA, TRA(n_{req}), to directly ask TRA(n_{pro}) for n_{pro}'s *r*-certificate. In Fig. 3, Step (1) and (2) indicate the process of *r*-certificate acquisition that is happening between n_{req} and n_{pro}. The detail of *r*-certificate acquisition are shown below:

- At n_{req} side
  - (1) TRA(n_{req}) sends a *CertRequest* message carrying its version number to TRA(n_{pro}) and then expects a reply message from it.
  - (2) If TRA(n_{req}) does not get any reply from TRA(n_{pro}) during a given period of time δ_{cert}, it will retry by sending a *CertRequest* message again to TRA(n_{pro}). If there is still no response from TRA(n_{pro}) after θ_{cert} (a preconfigured number of) retries, TRA(n_{req}) considers that there is no TRA(n_{pro}) at all and notifies n_{req} of the absence of TRA(n_{pro}). (This is referred to as Case 1.)
  - (3) If TRA(n_{req}) gets a reply from TRA(n_{pro}) within θ_{cert} retries, in the case that the reply is
    - (a) A Low version message,
      - (i) If, since the last *CNTER* resetting, it is the first time that TRA(n_{req}) realizes n_{pro} has a high version of TRA, TRA(n_{req}) increments *CNTER* and notifies n_{req} that it needs to be updated. (This is referred to as Case 2.)
      - (ii) Otherwise, TRA(n_{req}) pretends nothing happened.
    - (b) A High version message, TRA(n_{req}) notifies n_{req} that TRA(n_{pro}) is out-of-date. (This is referred to as Case 3.)
    - (c) A *CertReply* message,
      - (i) TRA(n_{req}) validates the RC(n_{pro}) contained in the message by the following steps:
        - (A) Decrypt the RC(n_{pro}), and retrieve the R. Recall $\text{RC}(n_i) = E_{\text{AK}}(R, H(R))$.
        - (B) Compute the hash digest of R, $H'(R)$.
        - (C) Retrieve $H(R)$ from $RC(n_{\text{pro}})$, and compare it with $H'(R)$. If they are equal, the validity test is passed, otherwise it fails.
      - (ii) If the validity check is passed, TRA(n_{req}) computes the trust of n_{pro}, based on the trust evaluation on n_{pro} in L_{val} (if any) and the reputation data retrieved from RC(n_{pro}), using its self-contained computation logic, and it then passes n_{req} the computation result. (This is referred to as Case 4.)
      - (iii) Otherwise, it considers that the *r*-certificate has been illegally modified by n_{pro} and notifies n_{req} of the situation. (This is referred to as Case 5.)

In Case 2, if *CNTER* is equal to or greater than a threshold value θ_{cnt}, n_{req} asks its cluster head for the latest version of the TRA; otherwise, it goes for another

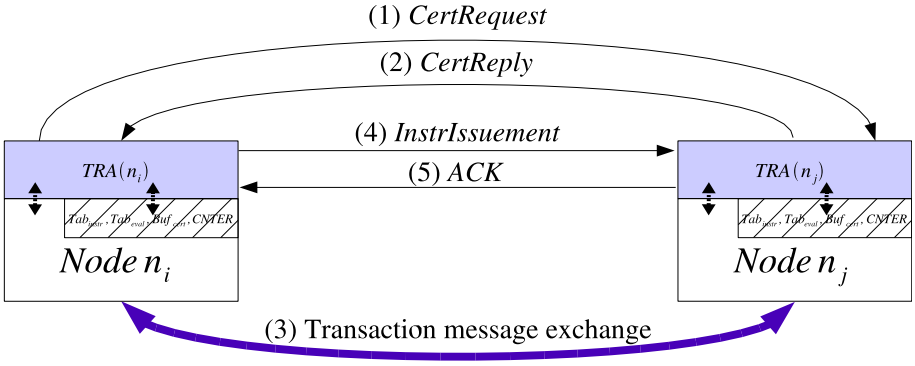

Fig. 3. The interaction between TRAs in the service-offering phase.

transaction partner instead of n_{pro}. In Case 4, based on the computation result, n_{req} makes the decision as to whether to start transaction TRAn$_\star$ with n_{pro}. In the other cases, what n_{req} is going to do depends on n_{req}'s local policy.

- At n_{pro} side
  (1) Upon receiving the *CertRequest* message from TRA(n_{req}), TRA(n_{pro}) retrieves the version number of TRA(n_{req}) from the message to check whether it itself is of the same version as TRA(n_{req}).
  (2) If their version numbers are consistent, TRA(n_{pro}) encapsulates the *r*-certificate of n_{pro}, RC(n_{pro}) (stored in Buf$_{cert}$), into a *CertReply* message and sends the message to TRA(n_{req}). (This is referred to as Case 6.)
  (3) Otherwise,
    (a) If the version number of TRA(n_{pro}) is higher than that of TRA(n_{req}), TRA(n_{pro}) sends a Low version message to TRA(n_{req}). (This is referred to as Case 7.)
    (b) Otherwise,
      (i) If, since the last *CNTER* resetting, this situation has ever happened with n_{req}, or it has already received a Low version message from n_{req}, TRA(n_{pro}) pretends nothing happened.
      (ii) Otherwise, TRA(n_{pro}) sends a High version message to TRA(n_{req}), increments *CNTER*, and notifies its host n_{pro} that it need to be updated. (This is referred to as Case 8.)

In Case 6, if TRA(n_{pro}) later receives n_{req}'s transaction request, it could accept or reject it. In Case 7, n_{pro} pretends nothing happened. In Case 8, if CNTER is equal to or greater than θ_{cnt}, n_{pro} asks its cluster head for the latest version of the TRA; otherwise, it pretends nothing happened.

4.5.2.2. t-Instrument issuance. *t*-Instrument issuance is triggered by TRA(n_{req}) in the post-transaction process right after the termination of TRAn$_\star$. In Fig. 3, Step (4) and (5) indicate the process of *t*-instrument issuance between n_{req} and n_{pro}. The detail of *t*-instrument issuance is presented as follows:

- At n_{req} side
  (1) Upon the termination of TRAn$_\star$, based on the QoS obtained from n_{pro} during TRAn$_\star$, n_{req} makes a trust evaluation on n_{pro} for every related context $C_\star$. A trust evaluation has the following form: (ID(n_{pro}),$C_\star$, $t_{req,pro}$,T) where T is the time when the evaluation is made. Then, n_{req} submits these trust evaluations to TRA(n_{req}).
  (2) For every trust evaluation (ID(n_{pro}),$C_\star$, $t_{req,pro}$,T) submitted by n_{req}, TRA(n_{req})
    (a) Updates Entry$_{eval}$(n_{pro},$C_\star$) with ($t_{req,pro}$,T) (if no such an entry, TRA(n_{req}) creates one first), and then
    (b) Generates a *t*-instrument TI(n_{req},n_{pro},$C_\star$,T) accordingly, sends it to TRA(n_{pro}), and then
    (c) Expects an ACK message from TRA(n_{pro}).
  (3) If TRA(n_{req}) does not receive from TRA(n_{pro}) the ACK message corresponding to a issued *t*-instrument in a certain period of time δ_{instr}, it will resend the *t*-instrument and expect an ACK message again. The *t*-instrument issuance can be retried maximally θ_{instr} (a pre-configured number of) times.
- At n_{pro} side
  (1) After receiving TI(n_{req},n_{pro},$C_\star$) from n_{req}, TRA(n_{pro}) verifies its validity in the same way as a *r*-certificate is validated.
  (2) If TI(n_{req},n_{pro},$C_\star$) is invalid, it is simply discarded by TRA(n_{pro}).
  (3) Otherwise, TRA(n_{pro}) first retrieves the time-stamp T from TI(n_{req},n_{pro},$C_\star$) and then looks up Entry$_{instr}$(n_{req},$C_\star$).
    (a) If Entry$_{instr}$(n_{req},$C_\star$) (shortly Entry$_{instr}$) exists,
      (i) In case of Entry$_{instr}$.TS $< T$, TRA(n_{pro}) first updates Entry$_{instr}$.instr and Entry$_{instr}$.TS respectively with TI(n_{req},n_{pro},$C_\star$) and T, and then sends an ACK message back to TRA(n_{req}), and afterwards resets Entry$_{instr}$.ACK to 1; or,

(ii) In case of $\text{Entry}_{\text{instr}}.\text{TS} = T$ and $\text{Entry}_{\text{instr}}.\text{ACK} < \theta_{\text{ack}}$, $\text{TRA}(n_{\text{pro}})$ sends an ACK message back to $\text{TRA}(n_{\text{req}})$ and then increments $\text{Entry}_{\text{instr}}.\text{ACK}$; or,

(iii) In all other cases, $\text{TRA}(n_{\text{pro}})$ discards $\text{TI}(n_{\text{req}},n_{\text{pro}},C_{\bigstar},T)$ and pretends nothing happened.

(b) Otherwise, $\text{TRA}(n_{\text{pro}})$ first creates in $\text{Tab}_{\text{instr}}$ a new entry $\text{Entry}_{\text{instr}}(n_{\text{req}},C_{\bigstar})$ with $\text{TI}(n_{\text{req}},n_{\text{pro}},C_{\bigstar})$ and T, and then sends an ACK message back to $\text{TRA}(n_{\text{req}})$.

4.5.2.3. r-Certificate issuance. r-Certificate Issuance is executed periodically by replica TRAs based on the *t*-instruments of their hosts. It involves two types of operations, computing reputation and generating *r*-certificate. For any node n_i, $\text{TRA}(n_i)$ computes the reputation of its host n_i based on the old *r*-certificate in Buf_{cert} and the *t*-instruments in $\text{Tab}_{\text{instr}}$ using its self-contained computation logic. Since *t*-instruments are context-specific, the computation result will not be a single value but a set of such values each of which represents n_i's reputation in a specific context at the time T when it is computed. Afterwards, $\text{TRA}(n_i)$ generates a *r*-certificate with time-stamp T for n_i based on the computation result of the previous step, and replaces the old *r*-certificate in Buf_{cert} with the new one, and then empties $\text{Tab}_{\text{instr}}$.

4.5.2.4. Trust management routine. Trust management routine is periodically carried out by every replica TRA to maintain the Tab_{eval} on its hosting node. Because of the temporalness of trust and the limit of local storage space, stale trust evaluation values should be removed from the Tab_{eval}. In each run of the routine, the replica TRA checks the difference between the current time and the time-stamp of each entry in the Tab_{eval}. If the difference is bigger than a threshold value δ_{valid}, the entry is removed.

5. Proof of correctness

In this section, we are going to present several lemmas and theorems about the ATRM, and give the proof of their correctness.

Lemma 5.1. *Every active node will eventually hold a replica of the latest TRA.*

Proof. During the *network initialization* phase, in the first stage, the AL broadcasts a TRA in the backbone network, and it is guaranteed that all living backbone nodes receive the TRA since the backbone network is connected despite backbone-node failure (by assumption); in the second stage, every cluster head (that is a backbone node) broadcasts the received TRA within its own cluster in the original network, but cluster members' receipt of the TRA can not be guaranteed due to node failure or cluster re-organization. Under these circumstances, after the *network initialization* phase, all living backbone nodes hold a replica of the most-recently launched TRA while some non-backbone nodes may not. For any non-backbone node n_i, its failure in receiving the latest TRA results in two possible cases: (1) n_i does not have a replica TRA at all; (2) n_i has an out-of-date TRA. For the first case, n_i is aware of its TRA absence and will actively ask its cluster head for the latest version of the TRA. In the second case, n_i will finally be notified of the staleness of its replica TRA when it does not succeed later in acquiring other nodes' *r*-certificates in the *service-offering* phase, and when the number of such failures is beyond θ_{cnt}, n_i will ask its cluster head for the latest version of the TRA. Therefore, n_i is able to have a replica of the latest TRA in either of the above two cases. Hence, sooner or later, every active node will eventually hold a replica of the latest TRA. □

Lemma 5.2. *All the trust evaluations on a node are aggregated on the node itself.*

Proof. In the post-transaction process of any transaction $\text{Tran}_{\bigstar}$, n_{req} commissions $\text{TRA}(n_{\text{req}})$ to issue n_{req} a *t*-instrument based on the QoS n_{req} provides during $\text{Tran}_{\bigstar}$. The *t*-instrument is then stored (in the $\text{Tab}_{\text{instr}}$) on n_{req} and managed by $\text{TRA}(n_{\text{req}})$. Therefore, a node will hold all the trust evaluations, in the form of *t*-instrument, from the nodes it ever had a transaction with as *provider*, and thus Lemma 5.2 holds its correctness. □

Theorem 5.1. *A node's reputation computed in ATRM reflects the overall trust evaluation of other nodes on the node.*

Proof. In ATRM, by choosing the accurate reputation model, the computed reputation can accurately reflect a nodes trustworthiness in a global view if sufficient individual trust evaluations of the node are provided. According to Lemma 5.2, for any node n_i, it holds all its *t*-instruments issued by other nodes. Therefore, the reputation computed by $\text{TRA}(n_i)$ using this full set of *t*-instruments will truly indicate the overall trust evaluation of n_i from the other nodes' point of view. Hence, Theorem 5.1 is correct. □

Theorem 5.2. *The reputation of a node can be directly acquired by any other node without network-wide flooding.*

Proof. The correctness of Theorem 5.2 is derived from the fact that every nodes reputation is locally stored by the node itself in the form of *r*-certificate. Remember that a node's reputation is objective and can be shared by all the other nodes in ATRM. For any transaction $\text{Tran}_{\bigstar}$, n_{req} knows the fact that n_{pro} locally stores its own *r*-certificate. Hence, n_{req} can get n_{pro}'s *r*-certificate simply by querying n_{pro} itself instead of by exhaustively asking every other node in the network. □

Theorem 5.3. *A node's locally-stored trust and reputation are readable to all the replica TRAs (of proper version) but are confidential to any node including the node itself.*

Proof. When a replica TRA generates a *t*-instrument/*r*-certificate, it encrypts the *t*-instrument/*r*-certificate with its secret key AK. By Lemma 5.1, all the replica TRAs are eventually derived from the same original TRA and share the same AK, and thus they are sooner or later able to successfully decrypt the *t*-instruments/*r*-certificates generated by each other. Since AK is securely kept only by replica TRAs, nodes are unable to read either their locally-stored *t*-instruments/*r*-certificates or received ones. In addition, by using strong cryptography algorithms and sufficiently long secret keys, successful attacks to AK via cryptanalysis and brute force guessing can be precluded. Hence, Theorem 5.3 holds. □

Theorem 5.4. *A node's unauthorized modification to its locally-stored trust and reputation information can be identified.*

Proof. Every *t*-Instrument/*r*-certificate is encrypted with the secret key of a replica TRA. According to the underlying network model, nodes are connected through secure links, and thus the data passed all the way to a replica TRA in the application layer is the original data sent from the sender node. Moreover, before *t*-instrument issuance and *r*-certificate provision happen between two nodes, their local replica TRA's are required to pass the consistency test. For any transaction *requester*, it has no reason to tamper with its received confidential *r*-certificates since they are critical for its decision-making, and it has no reason to tamper with the *t*-instruments it issues since it itself is the actual creator of these *t*-instruments. Under these circumstances, the failure of a replica TRA in validating the received *t*-instruments/*r*-certificates must be due to the host/sender nodes' unauthorized modification to the *t*-instruments/*r*-certificates. Hence, Theorem 5.4 holds its correctness. □

6. Complexity analysis

The complexity of ATRM is the sum of that of the *network initialization* phase and that of the *service-offering* phase. Because the *network initialization* phase (which is a broadcast process) is run only once at the beginning of the execution of ATRM (unless special security requirements are specified), in this section, we will analyze the complexity only of the service-offering phase. The complexity analysis is done under two assumptions: reliable transmission and transaction-qualified nodes.

According to Section 4.5.2, only *r*-certificate acquisition and *t*-instrument issuance involve message transmission during the *service-offering* phase, and for each *r*-certificate acquisition and *t*-instrument issuance process, there are two ATRM packets injected into the network respectively. In this case, the number of ATRM packets per transaction is four. Let $N_{\mathrm{TRAnsaction}}$ denote the number of transactions. Then, the message complexity of the *service-offering* will be exactly:

$$C_{\mathrm{msg}} = 4 \times N_{\mathrm{transaction}} \tag{5}$$

Based on this formula, we can see that the number of ATRM packets is directly proportional to the number of transactions during the *service-offering* phase but irrelevant to the number of nodes in the network. The solid lines in Figs. 4(a) and 5(a) clearly show this relation.

Now, let us analyze the average *r*-certificate acquisition delay, T_{delay}, which is defined as the average time difference between the time when a *CertRequest* message is sent from a transaction *requester* and the time when the corresponding *CertReply* message is successfully received by the *requester*. By Section 4.5.2, during a *r*-certificate acquisition process, a *CertRequest* message is first sent from the *requester* to the *provider*, and the *provider* then sends a *CertReply* message back to the *requester* after it finishes processing the received *CertRequest* message. Let us denote by T_{e2e} the average end-to-end delay of the underlying routing protocol and by T_{process} the average delay for processing a *CertRequest* message. Then, we have the following formula:

$$T_{\mathrm{delay}} = 2 \times T_{\mathrm{e2e}} + T_{\mathrm{process}} \tag{6}$$

For a particular WSN, T_{process} is a constant factor, and thus, T_{delay} actually depends on the routing protocol employed.

7. Performance evaluation

In this section, we evaluate the performance of ATRM in two aspects: (1) the message overhead added by ATRM to the underlying network during the *service-offering* phase; and (2) the average *r*-certificate acquisition delay, based on the experiments we have carried out using the Network Simulator, NS-2. Since the backbone network is used only for agent distribution in the *network initialization* phase, it was not simulated in our experiments.

During our experiments, we use a flat, rectangular area of 100 m × 100 m. Nodes are randomly deployed and are fixed throughout the simulation. They communicate with each other in the WSN using a radio of 60 m. Every node is transaction-qualified and is pre-assigned a TRA with a 64-bit secret key. The system parameters δ_{cert} and δ_{instr} are set to 20 s while θ_{cert} and θ_{instr} are configured to five. Some nodes spontaneously trigger transactions, each of which lasts for a randomly decided period of simulated time. During a transaction, the *requester* constantly sends 300-byte packets at the transmission rate of two packets per second. To have an average result, we run the simulation for each scenario five times. During each simulation run, we kept track of the number of ATRM packets and the average *r*-certificate acquisition delays. Based on the above setup, we conducted two sets of experiments. In the first set, where the number of nodes were set to 100 in order to simulate the high density of WSNs, we aimed at studying the performance of ATRM in the situations with a fixed number of nodes and a changing numbers of

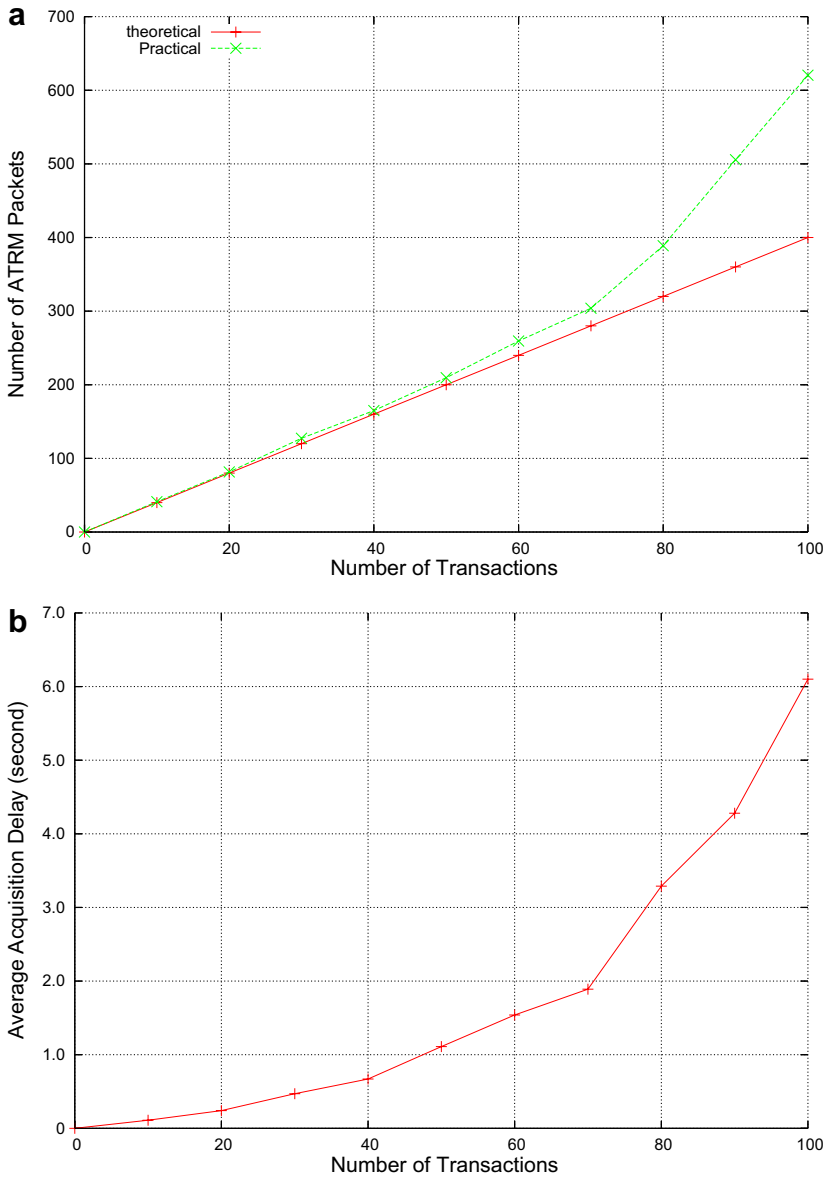

Fig. 4. Experiment set 1 with 100 nodes. (a) Number of ATRM packets Vs. Number of transactions. (b) Average acquisition delay Vs. Number of transactions.

transactions. In the second set, we investigated the performance of ATRM with a fixed number of transactions (50) and a varying number of nodes.

Fig. 4(a) shows the change in the number of ATRM packets in relation to the change in the number of transactions. The dashed line stems from experimental results while the solid one is based on theoretical analysis. Following either of these two lines, we can see that the number of ATRM packets rises as the number of transactions increases. With the increase of the number of transactions from 0 to 70, the two lines stay very close to each other, and the gap between them grows slowly. However, after the number of transactions is beyond 70, the gap between the two lines increases at a very fast pace. This situation is due to the impact of transactions on network performance. In fact, the moderate increase of traffic load can be comfortably adapted by the underlying network. Whereas, if the traffic load is beyond the limit that the network can adequately handle, network congestion and consequent packet loss will occur, furthermore, the heavier the traffic load, the more serious the situation. In our experiments, 70 transactions make the traffic load reach that limit, and therefore when the number of transactions is greater than 70, packet loss happens very often during the *r*-certificate acquisition and *r*-instrument issuance and thus cause the frequent failure of the two processes. By the definition of ATRM, when the two processes fail, transaction *requester* will retry. Frequent *r*-certificate acquisition and *r*-instrument issuance retries cause the injection of more ATRM packets into the network and thus widen the distance between the experimental results and theoretical analysis.

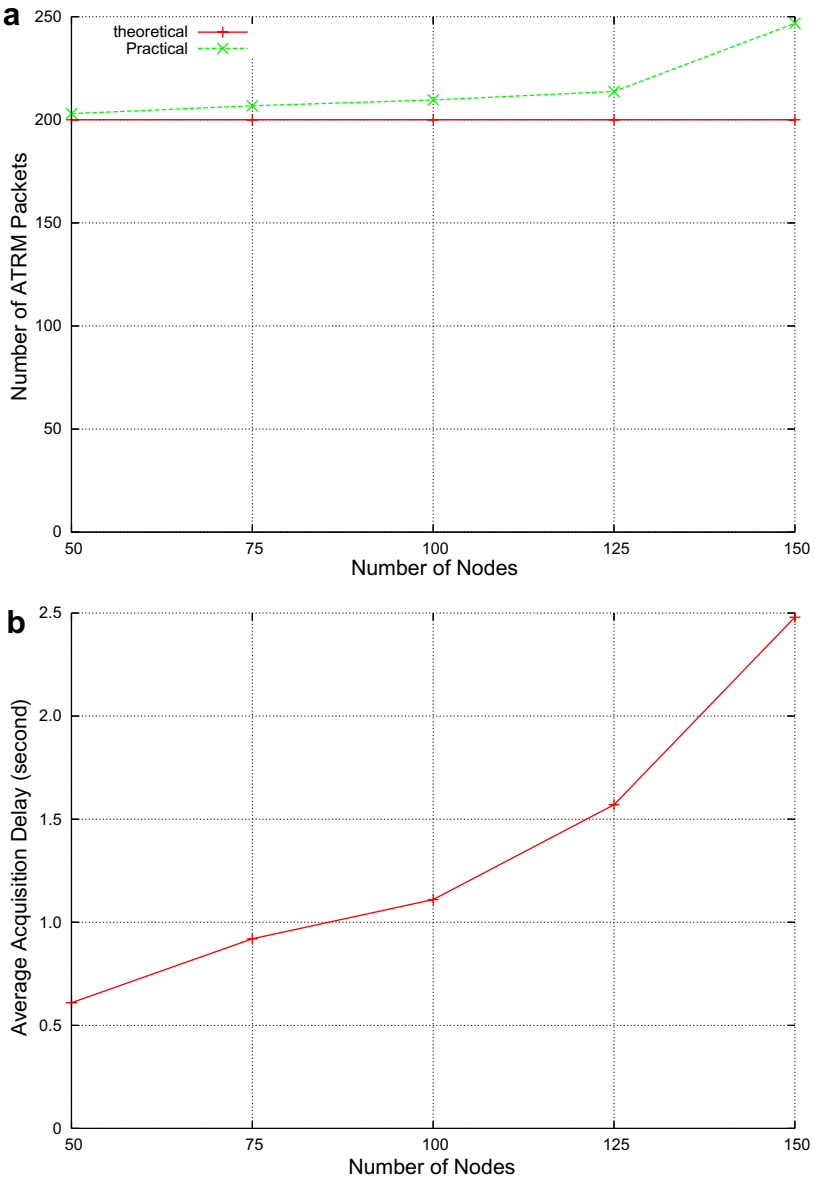

Fig. 5. Experiment set 2 with 50 transactions. (a) Number of ATRM packets Vs. Number of nodes. (b) Average acquisition delay Vs. Number of nodes.

Fig. 4(b) illustrates the average delay of *r*-certificate acquisition as a function of the number of transactions. In busy networks, network packets are enqueued before being processed, and the heavier the traffic load, the longer they stay in queue, and thus the longer the transmission delay. Thereby, the line goes up as the number of transactions increases.

Fig. 5(a) depicts the variation in the number of ATRM packets in relation to the change in the number of nodes when the number of transactions is fixed. The dashed line stems from experimental results while the solid one is based on theoretical analysis. Following the dashed line, we actually find that the number of ATRM packets increases as the number of nodes goes up. Clearly, these experimental results deviate from theoretical analysis. The reason for the difference is that per node throughput declines as the number of nodes increases [44] (note that channel capacity is fixed in our experiments). Poor throughput causes packet loss and retransmission, and thus results in the increase of the total number of ATRM packets.

Fig. 5(b) exhibits the average delay of *r*-certificate acquisition as a function of the number of nodes. As we explained previously, per node throughput goes down as the number of nodes increases in a wireless network with fixed channel capacity. Therefore, when the number of nodes becomes larger and larger, packets have to stay in queue for an increasingly longer period of time before they are processed. This makes the average delay of *r*-certificate acquisition rise.

8. Conclusion

Providing security in wireless sensor networks is a major requirement for the acceptance and deployment of these

networks. But achieving an acceptable level of security has been a difficult problem. The difficulty stems from the fact that wireless sensor networks have bandwidth and time constraints. In this paper, we have presented a novel trust and reputation management scheme that can protect the security of transacting entities. The scheme uses a localized trust and reputation management strategy, hence avoiding network-wide flooding. Our approach enables each node in the network to establish a trust value with other interacting entities with minimal communication cost and acquisition latency. Our extensive experimental results show that our scheme provides minimal overhead and can be adequately adopted for wireless sensor networks.

References

[1] A. Abdul-Rahman, S. Hailes, A distributed trust model, in: Proc. 1997 Workshop on New Security Paradigms, Langdale, UK, 1997, pp. 48–60.

[2] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in: Proc. 33rd Ann. Hawaii Int'l Conf. Syst. Sci. (HICSS 33), vol. 6, (2000) pp. 6007–6016.

[3] B. Alunkal, I. Veljkovic, G.V. Laszewski, K. Aminand, Reputation-based grid resource selection, in: Proc. Workshop Adaptive Grid Middleware, New Orleans, US, Sep. 2003. Available from <http://www-unix.mcs.anl.gov laszewsk/papers/vonLaszewski/> (reputation.pdf on May 06, 2005).

[4] F. Azzedin, M. Maheswaran. Towards trust-aware resource management in grid computing systems, in: Proc. Second IEEE/ACM Int'l. Symp. Cluster Comput. Grid (CCGRID'02), Berlin, Germany, May 2002, pp. 452–457.

[5] S. Banerjee, S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks, in: IEEE Infocom 2001, Anchorage, US, Apr. 2001, pp. 1028–1037.

[6] S. Basagni. Distributed clustering for ad hoc networks, in: Proc. 1999 Int'l. Symp. Parallel Arch., Alg. Netw. (ISPAN '99), Fremantle, Australia, June 1999, pp. 310–315.

[7] M. Blaze, J. Feigenbaum, J. Ioannidis, A.D. Keromytis, RFC 2704 - The KeyNote Trust Management System Version 2, Sep. 1999. Available from <http://www.faqs.org/rfcs/rfc2704.html> (on May 06, 2005).

[8] M. Blaze, J. Feigenbaum, J. Lacy. Decentralized Trust Management, in: Proceedings of 1996 IEEE Conference on Privacy and Security, Oakland, US, 1996, pp. 164–173.

[9] S. Buchegger, J. Boudec, A robust reputation system for P2P and mobile ad-hoc networks, in: Proc. Second Workshop the Economics of Peer-to-Peer Systems, Cambridge, US, June 2004. (Available from <http://www.eecs.harvard.edu/p2pecon/confman/papers/s2p2.pdf/> (on May 06, 2005).

[10] S. Buchegger, J.L. Boudec, Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks, in: Proc. Tenth Euromicro Workshop Parallel, Distr. Netw.-based Process., 2002, pp. 403–410.

[11] N. Budhiraja, K. Marzullo, Tradeoffs in implementing primary-backup protocols, in: Proc. Seventh IEEE Symp. Parallel Distr. Process., San Antonio, US, Oct. 1995, pp. 280–288.

[12] B. Christianson, W.S. Harbison, Why isn't trust transitive?, in: Proc. Security Protocols Int'l Workshop, University of Cambridge, 1996, pp. 171–176.

[13] P. Dewan, P. Dasgupta, Securing P2P networks using peer reputations: is there a silver bullet?, in: Proc. IEEE consumer communications and networking conference (CCNC 2005), Las Vegas, US, 2005. Available from <http://cactus.eas.asu.edu/partha/Papers-PDF/2005/Dewan-CCNC.pdf/> (on May 06, 2005).

[14] D. Gambetta. Can We Trust Trust?, in :D. Gambetta, (Ed.), Trust: Making and Breaking Cooperative Relations, Department of Sociology, University of Oxford, Electronic Edition, 1990 (pp. 213–237) (Chapter13).

[15] M. Gerla, J.T. Tsai, Multicluster, mobile, multimedia radio network, ACM-Baltzer J. Wirel. Netw. 1 (3) (1995) 255–265.

[16] V.S. Grishchenko, A fuzzy model for context-dependent reputation, in: Proceedings of Trust, Security and Reputation Workshop at ISWC 2004, Hiroshima, Japan, 2004. Available from <http://trust.mindswap.org/trustWorkshop/papers/1/> (s.pdf on May 06, 2005).

[17] M. Gupta, P. Judge, M. Ammar, A Reputation System for Peer-to-peer networks, in: Proc. 13th Int'l Workshop Netw. Oper. Syst. Support for Digital Audio and Video, Monterey, US, June 2003, pp. 144–152.

[18] P. Gupta, P.R. Kumar, The Capacity of Wireless Networks, IEEE Trans. Inform. Theory IT-46 (2) (2000) 388–404.

[19] Q. He, O.D. Wu, P. Khosla, SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks, in: Proceedings of IEEE Wireless Communications and Networking Conference 2004. Available from <http://www.wu.ece.ufl.edu/mypapers/WCNC04_incentive.pdf/> (on May 06, 2005).

[20] A. Herzberg, Y. Mass, J. Michaeli, D. Naor, Y. Ravid, Access control meets public key infrastructure, or: assigning roles to strangers, in: Proc. 2000 IEEE Symp. Security and Privacy, Berkeley, US, May 2000, pp. 2–14.

[21] R. Housley, W. Polk, W. Ford, D. Solo, RFC 3280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Apr. 2002. (Available at <http://www.ietf.org/rfc/rfc3280.txt/> (on May 06, 2005).

[22] R. Jurca, B. Faltings. An incentive compatible reputation mechanism, in: Proc. IEEE Int'l Conf. E-Commerce, June 2003.

[23] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks, in: Proc. 12th Int'l World Wide Web Conf., Budapest, Hungary, 2003, pp. 640–651.

[24] N. Li, J. Mitchell, RT, A Role-based Trust-management Framework, in: Proc. Third DARPA Inform. Surviv. Conf. Exposition (DISCEX'03), Apr. 2003, pp. 201–212.

[25] C.R. Lin, M. Gerla, Adaptive clustering for mobile networks, IEEE J. Select. Areas Commun. 15 (7) (1997) 1265–1275.

[26] J. Liu, V. Issarny, Enhanced reputation mechanism for mobile ad hoc networks, in: Proceedings of the Second International Conference on Trust Management, vol. 2995, Mar. 2004, pp. 48–62.

[27] Z. Liu, A.W. Joy, R.A. Thompson, A dynamic trust model for mobile ad hoc networks, in: Proc. Tenth IEEE Int'l Workshop Future Trends Distr. Comput. Syst. (FTDCS'04), Suzhou, China, May 2004, pp. 80–85.

[28] D.H. McKnight, N.L. Chervany, The Meanings of Trust Technical Report 94-04, Carlson School of Management, University of Minnesota, 1996.

[29] P. Michiardi, R. Molva, Core: A collaborative reputation mechanism to enforce node cooperation in mobile AD HOC networks, in: Proc. Sixth IFIP Commun. Multimedia Secur. Conf., Portorosz, Slovenia, 2002, pp. 107–121.

[30] B.A. Misztal, Trust in Modern Societies: The Search for the Bases of Social Order, Polity Press, 1995.

[31] L. Mui, M. Mohtashemi, A. Halberstadt, A computational model of trust and reputation, in: Proc. 35th Hawaii Int'l Conf. Syst. Sci., 2002, pp. 188–196.

[32] T.G. Papaioannou, G.D. Stamoulis, Effective use of reputation of peer-to-peer environments, in: Proceedings of IEEE/ACM CCGRID 2004, GP2PC workshop, Chicago, U.S., Apr. 2004. (Available from <http://nes.aueb.gr/publications/2004.p2p_policies.GP2PC.pdf/> (on May 06, 2005).

[33] A.A. Pirzada, C. McDonald, Establishing trust in pure ad-hoc networks, in: Proc. 27th Conf. Australasian Computer Sci., ACM Int'l Conf. Proc. Series, Dunedin, New Zealand, 2004, pp. 47–54.

[34] S. Pleisch, A. Schiper, Modeling fault-tolerant mobile agent execution as a sequence of agreement problems, in: Proc. 19th IEEE Symp. Reliable Distr. Syst., Nuremberg, Germany, Oct. 2000, pp. 11–20.
[35] K. Ren, T. Li, Z. Wan, F. Bao, R.H. Deng, K. Kim, Highly reliable trust establishment scheme in ad hoc networks, Eslevier J. Comput. Netw., Apr. 2004, pp. 687–699.
[36] K. Rothermel, M. straber, A fault-tolerant protocol for providing the exactly-once property of mobile agents, in: Proc. 17th IEEE Symp. Reliable Distr. Syst., 1998, pp. 100–108.
[37] T. Sander, C. Tschudin, Protecting mobile agents against malicious hosts, in: Mobile Agents and Security, Lecture Notes in Computer Science, vol. 1419, Heidelberg, Germany, 1998. pp. 44–60.
[38] F.B. Schneider, Towards fault-tolerant and secure agentry, in: Proc. 11th Int'l Workshop on Distributed Algorithms, Saarbrucken, Germany, Sep. 1997, pp. 1–14.
[39] D. Scott, A. Beresford, A. Mycroft, Spatial security policies for mobile agents in a sentient computing environment, in: Proceedings of FASE 2003, Lecture Notes in Computer Science, vol. 2621, Warsaw, Poland, Apr. 2003, pp. 102–117.
[40] Y. Tahara, A. Ohsuga, S. Honiden, Mobile agent security with the IPEditor development tool and the mobile UNITY language, in: Proc. Fifth Int'l Conf. Autonomous Agents, Montreal, Canada, 2001, pp. 656–662.
[41] J. Tardo, L. Valente, Mobile agent security and telescript. n Proceedings of the 41st Int'l Conf. IEEE Comput. Soci. (CompCon '96), pages 58–63, Feb. 1996.
[42] G. Theodorakopoulos, J.S. Baras. Trust evaluation in ad-hoc networks, in: Proc. 2004 ACM Workshop on Wirel. Security, Philadelphia, U.S., 2004, pp. 1–10.
[43] Y. Wang, J. Vassileva, Trust and reputation model in peer-to-peer networks, in: Proc. Third Int'l Conf. Peer-to-Peer Comput. (P2P'03), Linkoping, Sweden, Sep. 2003, pp. 150–157.
[44] K. Xu, X. Hong, M. Gerla, Landmark routing in ad hoc networks with mobile backbones, J. Parallel Distri. Comput. 63 (2) (2003) 110–122.
[45] P.R. Zimmermann, The Official PGP Users Guide, MIT Press, 1995.

Azzedine Boukerche is a Full Professor and holds a Canada Research Chair position at the University of Ottawa. He is the Founding Director of PARADISE Research Laboratory at uOttawa. Prior to this, he held a Faculty position at the University of North Texas, USA, and he was working as a Senior Scientist at the Simulation Sciences Division, Metron Corporation located in San Diego. He was also employed as a Faculty at the School of Computer Science McGill University, and taught at Polytechnic of Montreal. He spent a year at the JPL/NASA-California Institute of Technology where he contributed to a project centered about the specification and verification of the software used to control interplanetary spacecraft operated by JPL/NASA Laboratory.

His current research interests include wireless ad hoc and sensor networks, wireless networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, peformance evaluation and modeling of large-scale distributed systems, distributed computing, large-scale distributed interactive simulation, and parallel discrete event simulation. Dr. Boukerche has published several research papers in these areas. He was the recipient of the Best Research Paper Award at IEEE/ACM PADS'97, and the recipient of the 3rd National Award for Telecommunication Software 1999 for his work on a distributed security systems on mobile phone operations, and has been nominated for the best paper award at the IEEE/ACM PADS'99, ACM MSWiM 2001, and ACM MobiWac 2006.

Dr. A. Boukerche is a holder of an Ontario Early Research Excellence Award (previously known as Premier of Ontarion Research Excellence Award), Ontario Distinguished Researcher Award, and Glinski Research Excellence Award. He is a Co-Founder of QShine Int'l Conference, on Quality of Service for Wireless/Wired Heterogeneous Networks (QShine 2004), served as a General Chair for the 8th ACM/IEEE Symposium on modeling, analysis and simulaltion of wireless and mobile systems, and the 9th ACM/IEEE Symposium on distributed simulation and realt time-application, a Program Chair for ACM Workshop on QoS and Securuty for Wireless and Mobile networks, ACM/IFIPS Europar 2002 Conference, IEEE/SCS Annual Simulation Symposium ANNS 2002, ACM WWW'02, IEEE MWCN 2002, IEEE/ACM MASCOTS 2002, IEEE Wireless Local Networks WLN 03-04; IEEE WMAN 04-05, ACM MSWiM 98-99, and TPC member of numerous IEEE and ACM sponsored conferences. He served as a Guest Editor for the Journal of Parallel and Distributed Computying (JPDC) (Special Issue for Routing for mobile Ad hoc, Specail Issue for wireless communication and mobile computing, Special Issue for mobile ad hoc networking and computing), and ACM/kluwer Wireless Networks and ACM/Kluwer Mobile Networks Applications, and the Journal of Wireless Communication and Mobile Computing.

He serves as Vice General Chair for the 3rd IEEE Distributed Computing for Sensor Networks (DCOSS) Conference 2007, and as Program Co-Chair for Globecom 2007 Symposium on Wireless Ad Hoc and Sensor Networks. Dr. A. Boukerche serves as an Associate Editor for ACM/Kluwer Wireless Networks, Wiley In't Journal of Witeless Communication and Mobile Computing, the Journal of Parallel and Distributed Computing, and the SCS Transactions on simulation. He also serves as a Steering Committee Chair for the ACM Modeling, Analysis and Simulation fo Wireless and Mobile Systems Symposium, the ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks and the IEEE Distributed Simulation and Real-Time Applications Symposium (DS-RT).

Li Xu has received his MSc Thesis at the University of Ottawa His MSc Thesis has been focusing on security for Ad hoc routing protocols and Trust Management for MANETs. For further details on his work on Security and Trust Management for MANET and Sensor Networks, please refer to his MSc Thesis work under the supervision of Prof. A. Boukerche.

Khalil El-Khatib is an Assistant professor at the University of Ontario. he received his PhD degree from the University of Ottawa, and he has been working as a Research Officer at the National Research Council of Canada (NRC). His researc areas of interests are wireless ad hoc sensor networks, Network Security, and Privacy and trust for distributed and mobile systems.