Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2016, Article ID 3136853, 18 pages
http://dx.doi.org/10.1155/2016/3136853

Research Article

SEMAN: A Novel Secure Middleware for Mobile Ad Hoc Networks

Eduardo da Silva[1,2] and Luiz Carlos Pessoa Albini[1]

[1]*Department of Informatics, Federal University of Parana (UFPR), 81531970 Curitiba, Brazil*
[2]*Department of Informatics, Catarinense Federal Institute (IFC), 89245000 Araquari, Brazil*

Correspondence should be addressed to Eduardo da Silva; eduardo@ifc-araquari.edu.br

Received 28 December 2015; Revised 18 April 2016; Accepted 26 April 2016

Academic Editor: Tzonelih Hwang

As a consequence of the particularities of Mobile Ad Hoc Networks (MANETs), such as dynamic topology and self-organization, the implementation of complex and flexible applications is a challenge. To enable the deployment of these applications, several middleware solutions were proposed. However, these solutions do not completely consider the security requirements of these networks. Based on the limitations of the existing solutions, this paper presents a new secure middleware, called Secure Middleware for Ad Hoc Networks (SEMAN), which provides a set of basic and secure services to MANETs aiming to facilitate the development of distributed, complex, and flexible applications. SEMAN considers the context of applications and organizes nodes into groups, also based on these contexts. The middleware includes three modules: service, processing, and security. Security module is the main part of the middleware. It has the following components: key management, trust management, and group management. All these components were developed and are described in this paper. They are supported by a cryptographic core and behave according to security rules and policies. The integration of these components provides security guarantees against attacks to the applications that usethe middleware services.

1. Introduction

Mobile Ad Hoc Networks (MANETs) are very attractive in several scenarios, as [1] soldiers carrying out information on a battlefield; people sharing information during a meeting; attendees using notebooks in an interactive conference; rescuers working after disasters. MANETs are dynamically established without any fixed infrastructure or centralized administration, and their operation is held by the nodes themselves [2].

On the other hand, these characteristics also impose several challenges, and the major ones are related to security. In addition to conventional wireless communication problems, the dynamic topology facilitates action of adversaries, making MANETs susceptible to both passive and active attacks [2]. In a passive attack, an unauthorized adversary tries to discover or to use system information without interaction with the network, while in an active one, the adversary tries to break into the system aiming to affect its operation [3]. Due to these particularities, the development of applications for MANET is not an easy task [4].

In general networks, to support the resolution of heterogeneity, scalability, and resource sharing problems and to allow the implementation of more complex and flexible applications, middleware solutions have been successfully used [5]. These solutions aim to provide interoperability and other services, as distribution of functionality, scalability, load balancing, and fault tolerance [6].

Several middleware solutions have also been proposed to support applications and services on MANETs. These solutions are message-oriented [7] and were classified, in a previous work [4], on tuple space-based, P2P-based, context-based, cross-layer, and application-oriented solutions. A complete analysis and comparison of these middleware solutions can be found in [4].

To support MANETs characteristics, a middleware must be lightweight in terms of the amount of processing, memory, and bandwidth consumption, to maintain its overhead as

small as possible. Further, it must self-adapt to the dynamic environment and handle unpredictable message loss. Considering the security requirements of MANETs and since the middleware handles all the communication between a client and an application, it must also considers the security requirements [6]. However, middleware solutions do not consider, or consider only partially, the security requirements of the MANETs. As a consequence, security can be considered the main drawback of the solutions reported on [4]. Thus, the need for developing a middleware arises which considers the security challenges of MANETs and contains a set of components to provide secure services to applications.

This paper proposes a new Secure Middleware for Mobile Ad Hoc Networks, called SEMAN (*Secure Middleware for Mobile Ad Hoc Networks*). The middleware is based on groups, which are formed considering the context information. Group members exchange information which can be used on decision making and services provisioning. SEMAN has a security module which aims to ensure the system confidentiality and the resistance to following malicious attacks: selfish, byzantine, impersonation, and Sybil. The contributions of this paper are the new architecture of a secure middleware for MANETs; the design of a context-based group management and definition of strategies to the secure group communication; the integration of trust management and evaluation services; and an identity-based key management scheme integrated with the SEMAN cryptographic core.

The remaining of the paper is composed of six sections. Section 2 presents the network and the attack model. Section 3 introduces SEMAN: Secure Middleware for Mobile Ad Hoc Networks. Section 4 details the security model of SEMAN and how it can ensure security for middleware clients. Section 5 presents a scheme for secure group communication based on key agreements. Section 6 presents the integration of the security components to provide security to applications on different scenarios. Finally, Section 7 presents the conclusions and some research directions.

2. Network and Attack Model

The proposed middleware considers an asynchronous network composed of n mobile nodes, denoted by $N_1, \ldots, N_n$. The description of SEMAN considers the notation presented as follows:

$\mathbb{G}_1$: cyclic additive group with order p,

$\mathbb{G}_2$: cyclic multiplicative group with order p,

e: bilinear pairing in which $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$,

$H_1(x)$: hash function in which $H_1(x) = \{0,1\}^* \to \mathbb{G}_1^*$,

$H_2(x)$: hash function in which $H_2(x) = \mathbb{G}_2 \to \mathbb{Z}_p^*$,

$N_{\mathcal{F}}$: founder nodes,

N_i: identification of node i,

SK_i: private key of node i,

PK_i: private key of node i,

MSK: master private key of system,

MPK: master public key of system,

MSK_i: share of master private key hold by node i.

It is assumed that only trust nodes participate in group initialization phases.

SEMAN aims to protect the network against some malicious attacks as selfish, byzantine, impersonation, and Sybil. Even though SEMAN may be effective against other attacks, they were not considered in this paper.

(i) Selfish Attacks. A node presents a selfish behavior either as a consequence of a malicious and intentional act or to save its own resources. But regardless the reason, the selfish behavior may compromise the network activities and any decision-making scheme which needs the cooperation of nodes. To guarantee the security against a selfish behavior, SEMAN group operations are structured considering the secret sharing (t over n) technique, in which $n - (t + 1)$ nodes may be unavailable, or present a selfish behavior, and the system is able to attend the requests. Further, trust management component provides information about the behavior of nodes in a context. Thus, if a node is selfish and does not participate on group activities, the other nodes will be aware of this behavior through the trust management component.

(ii) Byzantine Attacks. A malicious node may perform a byzantine attack against the system, issuing false information or making decisions on behalf of a group. Thus, a byzantine attack may compromise the reliability of the middleware operations. The strategy to organize nodes into groups using secret sharing technique t-over-n increases the system protection against these attacks. Then, a malicious node must compromise, at least, other t nodes to perform any malicious activity on behalf of a group, making its actions more difficult and limited. Also, similar to selfish attack, the trust management provides ways for nodes to inform other ones if they detect some byzantine behavior. Thus, based on trust information, byzantine nodes can be isolated.

(iii) Impersonation Attacks. An attacker may steal the identity of a node and compromise the system reliability issuing false information on behalf of this node. A key management service is implemented to prevent this kind of attack against the system. All secure services provided by the middleware use cryptography. By using the key management, the middleware ensures that an identity belongs to the node that is using it. Thus, an attacker needs to compromised the entire key management component to be success on this kind of malicious action. Also, a secure communication service provided by the group management is implemented which increases the reliability of SEMAN against impersonation attacks, ensuring that only members of a closed group are able to decrypt a message sent to this group.

(iv) Sybil Attack. In a Sybil attack, a malicious node creates a false identity and gets the authorization of other nodes to have this false identity accepted in the system. Then, system reliability is affected, since a unique node may perform several activities on behalf of a group. Similar to impersonation attack, key management helps to prevent the action of a Sybil attacker. As the identity of a node is certified by the key management, it is necessary that an attacker compromises the

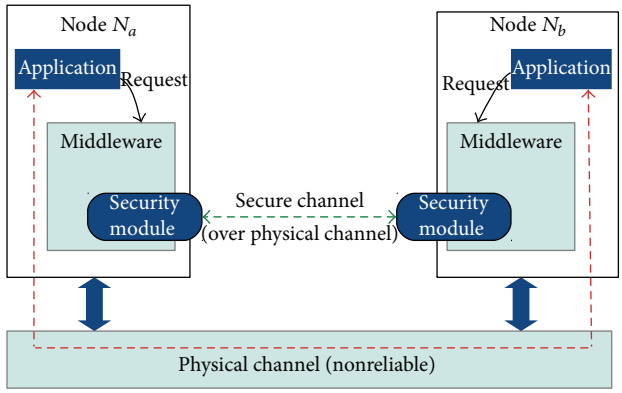

Figure 1: Reliable communication using a secure middleware.

key management scheme to be able to create a false identity and issue a valid public/private key pair to this new identity. Also, the secure communication of the group management component increases the reliability of SEMAN against this attack. As the secure communication service ensures that only the members of a closed group are able to decrypt a message sent to a group, it prevents a node from creating a false identity and from using this identity to receive messages sent to a group that it is not a member of.

This section presented the network characteristics addressed in this work and the attack model considered. Four kinds of attacks and how they may compromise the network behavior, performance, and operations were described.

3. Secure Middleware for Mobile Ad Hoc Networks

This section describes the SEMAN, the new context-based middleware that utilizes a group approach to support decision making related to security. Based on [4], context-based solutions are more suitable for MANETs. Further, the group approach facilitates the nodes organization on the contexts and security-related decision making. Figure 1 illustrates how applications may use the middleware to perform a reliable communication over an unreliable physical channel.

SEMAN provides support to secure and reliable communications between multiple nodes in scenarios susceptible to malicious attacks. It is composed of distributed modules and a set of group-based cryptographic operations. To support middleware operations, nodes with similar requirements form groups. These groups, called context groups, are self-organized and dynamically formed with no user interaction, considering only applications profiles and requirements. Services are provided and utilized by the applications in a context and, consequently, they are made available to nodes that belong to groups of this context.

SEMAN is composed of a communication interface, a catalog, and three modules: services, processing, and security, as illustrated in Figure 2. Applications requests may be sent either to the middleware or to lower layers. Without loss of generality, only the former scenario is considered in this paper. All message exchanges between the middleware and the applications are performed by using the communication interface, which classifies messages and deliveries them to the correct module or application.

The catalog is composed of a nonvolatile memory and is responsible for keeping all pending requests and security information about applications and nodes, like cryptographic keys, trust information, credential, and so forth. It is important to ensure the resistance in, at least, three scenarios: (i) node crash; (ii) network disconnection; and (iii) long delays in service provisioning. These situations may result either from a malicious action or from the dynamic behavior of MANETs.

The next subsections present the main characteristics and functionalities of the three modules and their components.

3.1. Services Module. The services module is responsible for keeping a list and details of all services and applications hosted by the node to other nodes. It encompasses the resource management, mobility management, and distributed storage. All these components are accessed by internal and external applications.

3.1.1. Resource Management. A service that provides information about location and availability of resources, as nodes, remote services, and contents, is very important for MANETs [8]. This service must consider the following restrictions: (i) mitigate the communication overhead, avoiding unneeded updates about available resources; (ii) be independent of nodes geographical position; and (iii) be independent of routing protocol. This component must consider the resources discovery and allocation, as well as their location management.

Resource management must offer, at least, four subcomponents, as depicted in Figure 3: allocation, registration, discovery, and location of resources. Each subcomponent requests and provides information to processing and security modules. For example, the security module provides information about nodes and applications authorization, while the resource allocation subcomponent provides information about the use of resources to the processing module.

Information about resources is locally stored and available to all local applications that use the middleware services. Also, this information can be available to other nodes, considering their context and permissions. The control access is maintained by security module and is based on context group formation.

3.1.2. Mobility Management. This component is particularly important, since mobile nodes can move and change their geographical positions unpredictably, affecting the performance of distributed applications. Besides nodes mobility, this service must consider the applications mobility, which can migrate from a node to another. To provide an effective service to applications, it contains three subcomponents, as depicted in Figure 4: location management, transfer management, and disconnection management.

Location management must provide information about nodes physical location to applications. Thus, it is assumed

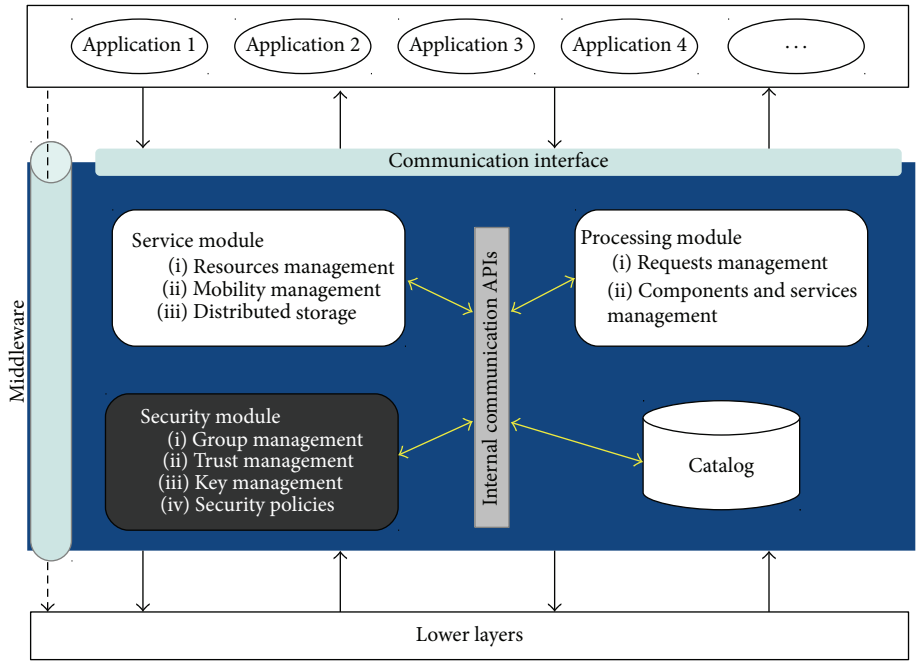

Figure 2: Architecture of the secure middleware.

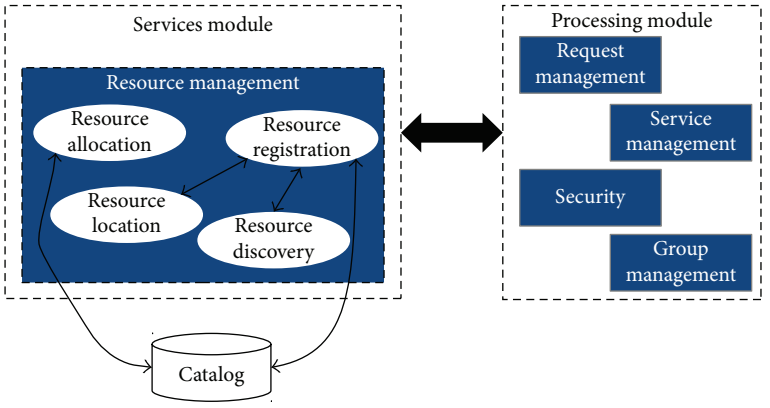

Figure 3: Components of resource management module.

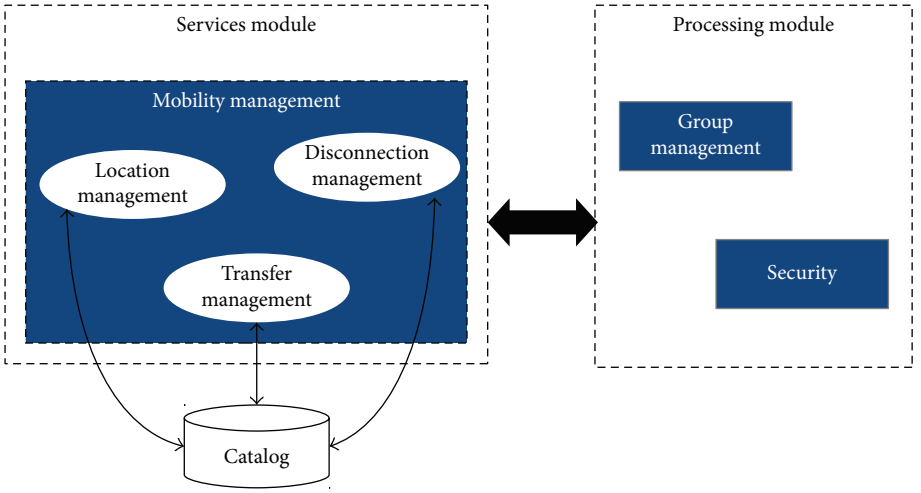

Figure 4: Components of mobility management module.

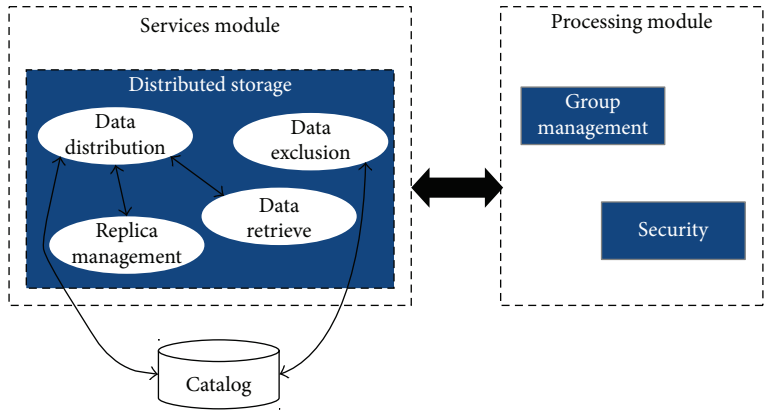

Figure 5: Components of distributed storage component.

that members of a group keep their location up-to-date in this group. Transfer management must allow mobile applications to keep the connection during a migration. It aims to mitigate the delay of applications transfers and to eliminate applications losses resulting from the migration. Finally, disconnection management must provide information about the reachability or disconnections of nodes that provide services through SEMAN.

3.1.3. Distributed Storage. This component allows nodes to store their information in a distributed, secure, dynamic, and self-organized way. It does not depend on any specific node availability and must be highly resistant to malicious attacks. Its main objective is to distribute context information to nodes of a group related to this context. This context information is fragmented into the network, and the absence of some nodes should not affect the stored data retrieval.

It is composed of four subcomponents, illustrated in Figure 5: data distribution, data retrieve, replica management, and data exclusion. All these components have a relationship with security and group management modules.

Data distribution is responsible for disseminating information to remote nodes. Data retrieve handles the access requests and locates remote data. Replica management is responsible for keeping enough active replicas to ensure the information availability and the data consistency. Exclusion data guarantees that when requested, data will be excluded from all remote nodes in which it is stored.

3.2. Processing Module. Processing module is responsible for keeping the central operation of SEMAN. It is composed of requests management and services and components management.

3.2.1. Requests Management. This component is responsible for keeping a registration of all services requests made to the middleware. It keeps the registration of both pending and attended requests.

An application is able to use, simultaneously, one or more services provided by the middleware. Due to the dynamic characteristics of MANETs, applications may not be aware of

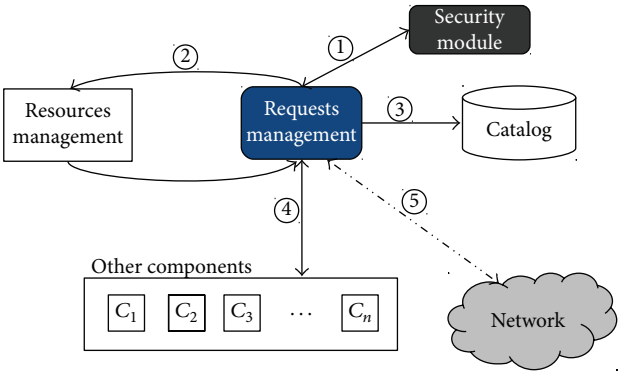

Figure 6: Services requests.

which services are being provided in a given time or which nodes are hosting these services. Figure 6 depicts how a service request is operated by SEMAN. Upon receiving a request, the request management component gets the security parameters with the security module. Then, it must check the availability of the requested service with the resource management component. If the service is provided by the middleware, it stores the information about the request into the catalog, performs the required communications with other components, and sends the request to the corresponding nodes.

As services may be provided by more than one node, SEMAN might

(1) request the service to all nodes that provide it, increasing the service availability and reducing the reply time;

(2) distribute the requests among nodes that provide it, making a load balance; or

(3) choose the more reliable node based on previous experiences.

During the service provisioning, the middleware may provide mechanisms to prevent malicious attacks. It must authenticate and authorize applications. Also, all messages exchanged with the middleware must be ciphered, to prevent eavesdropping.

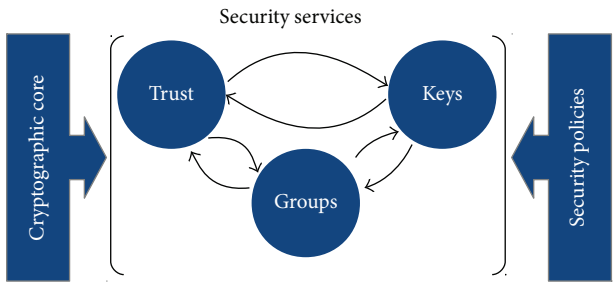

Figure 7: Security module diagram.

3.2.2. Services and Components Management. This component has a simple but essential function to the middleware operation. It is responsible for keeping a registration of all services and components provided by SEMAN. When a user wants to make a service available, through the middleware, this services must be previously registered. All relevant information of the new service, as security policies and context, must be stored into catalog. Then, other nodes can be informed about the availability of the new service.

This component must provide primitives to the registration of new services, as well the query of provided services. Similarly, for each registered component, it is necessary to store information about access strategies and services requirements.

3.3. Conclusion. This section presented an overview of SEMAN, its modules, and main characteristics. All next section will detail the security operations and module and how they will provide security to the middleware applications.

4. Security Module

This module is the central point of SEMAN. Its components, as depicted in Figure 7, include key management, trust management, and group management. These components operate with cryptographic operations and security policies components, which provide basic security primitives to the module.

Security services use a context-based group approach. All management operations and decision making are based on information provided by other members of the context group. Thus, nodes cooperate among themselves to increase the reliability of available services. However, the use of all components is not mandatory.

4.1. Cryptographic Core. To ensure that messages are not prone to eavesdropping, all messages must be ciphered. Though any cryptographic mechanism can be used, identity-based ones seem to be more suitable for MANETs [9]. Symmetric schemes impose a high cost to manage pairwise secret keys, and when compared with traditional certificate-based asymmetric schemes, identity-based cryptography (IBC) presents at least three advantages [10]:

(1) does not require certificates, mitigating certificates storage, distribution, and verification cost;

(2) makes easy noninteractive key agreement, reducing communication and processing overhead;

(3) removes the requirement of destination public keys authentications before message sending.

Another advantage to use IBC on MANETs is that they have a simple key management and a reduced storage cost, if compared with other methods. On IBCs, the identity of a user, as e-mail or IP, is used to derive the node public key. Thus, all nodes are able to disclose the public key of other nodes without data exchange. However, IBCs present a drawback. Private key is generated and available by an entity known as Private Key Generator (PKG). This characteristic imposes an implementation challenge, since PKG can be a single point of failure. To mitigate the impact of a central PKG, in this work the PKG is distributed over the network.

4.2. Cryptographic Operations. SEMAN considers the use of identity-based cryptography. Any IBC can be used, depending on middleware requirements. Without loss of generality, the Boneh and Franklin scheme is employed [11].

The main algorithms to perform cryptographic operations and to support the secure communication between nodes are composed of configuration and extraction and encryption and decryption. The former two are detailed on Section 4.4, which discusses the key management, since they are related to system initialization and key issuing.

The encryption and decryption algorithms are presented as follows. Let $k \in \mathbb{Z}^+$ be a security parameter from security module and $\mathbb{G}$ a generator of BDH parameter.

(1) *Encryption*: to encrypt M using a public key of node i, follow the next steps:

(a) calculate $\mathrm{PK}_i = H_1(N_i)$;

(b) choose a random $r \in \mathbb{Z}_q^*$;

(c) generate the encrypted text $C = \langle rP, M \oplus H_2(g_i^r) \rangle$ in which $g_i = \widehat{e}(N_i, \mathrm{PK}_i) \in \mathbb{G}_2^*$.

(2) *Decryption*: let $C = \langle U, V \rangle$ be the encrypted text using the public key of node i. To decrypt the message the private key SK_i is necessary. The following formula shows as text may be decrypted:

$$V \oplus H_2\left(\widehat{e}\left(\mathrm{SK}_i, U\right)\right) = M. \quad (1)$$

The proof of these algorithms can be found in [11].

4.3. Trust Management. Though cryptography may be used to ensure communication security, it does not provide information about the reliability of the nodes [12]. Further, many cryptographic mechanisms, such as key management [13, 14], rely on some degree of preestablished trust between nodes. However, trust in any kind of open network is very difficult to be valued and has received a lot of attention from the security community [15].

In a previous work [16, 17], TRUE was presented to evaluate the trust between pairwise communicating nodes. TRUE uses the concept of trust chains formed between nodes by

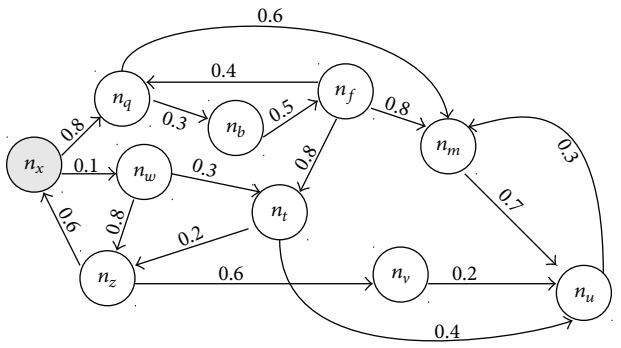

Figure 8: Example of trust chain G^x_{tr} from node n_x.

direct monitoring and recommendations of physical neighbors. It supports applications in an autonomous and dynamic way, while keeping the ability to resist malicious attacks. In this approach, each node creates in a self-organized strategy a trust network based on context, to provide trust information represented by a direct graph $G_{\text{tr}} = (V_{\text{tr}}, E_{\text{tr}})$, in which vertexes V_{tr} are the nodes and edges E_{tr} are the trust relationships between them. The trust network, or trust graph, contains all trust information which a node has about other nodes in a context. This information, or evidences, is gathered via direct interaction or by recommendations, considering the system security policies. The trustworthiness of a node is always locally computed, with no kind of message exchange, based on the trust network. In the next subsection are briefly presented the operation of the TRUE and its procedures and an evaluation of the proposed service.

4.3.1. Building Context-Based Trust Networks. When joining the system, each node N_i creates its own trust network $G^i_{\text{tr}} = (V^i_{\text{tr}}, E^i_{\text{tr}})$ in a self-organized way. Initially, nodes have knowledge only about nodes with which they have direct trust relations, and only such data is stored in the trust network. Then, in predetermined time intervals (ΔT_{ex}), nodes exchange trust evidences with their physical neighbors, propagating trust values through the network in an epidemic behavior [18, 19].

During trust information exchange, each node evaluates the relevance of the received evidences by calculating the trustworthiness of the sender. Then, it decides whether it accepts or not such evidences, based on local policy rules. If it accepts the trust evidences, then it incorporates the received information on its context-based trust network. Otherwise, trust evidences are discarded.

4.3.2. Trust Evaluation. To evaluate the trust on node N_u, node N_x must either have a direct connection with node n_u in G^x_{tr} or it finds at least one trust chain (TC) from N_x to N_u in G^x_{tr}. Trust chains represent a transitive trust from N_x to N_u. The trust network graph G^x_{tr} is depicted in Figure 8. As node N_x can find several different trust chains between itself and N_u in G^x_{tr}, each chain is denoted as $\text{TC}^i_{(N_x,N_u)}$.

If N_x has a direct trust with N_u, only this value is considered on trust evaluation. Otherwise, it tries to find a trust path in G^x_{tr}. Upon finding a chain, node N_x must compute its trust. Consider N_1 to N_m as the m intermediary nodes in the ith trust chain, denoted as $\text{TC}^i_{(N_x,N_u)}$; (2) estimates the trustworthiness of $\text{TC}^i_{(N_x,N_u)}$:

$$\text{TC}^i_{(N_x,N_u)} = \text{TV}_{(N_x,N_1)} \times \prod_{j=1}^{m-1} \text{TV}_{(N_j,N_{j+1})} \times \text{TV}_{(N_m,N_u)}. \quad (2)$$

Returning to Figure 8, there are several chains between n_x and n_u, for example,

(1) chain $(N_x \rightarrow N_q \rightarrow N_m \rightarrow N_u)$, trust chain value $\text{TC}^1_{(N_x,N_u)} = 0.8 \times 0.6 \times 0.7 = 0.336$;

(2) chain $(N_x \rightarrow N_q \rightarrow N_b \rightarrow N_f \rightarrow N_m \rightarrow N_u)$, trust chain value $\text{TC}^2_{(N_x,N_u)} = 0.8 \times 0.3 \times 0.5 \times 0.8 \times 0.7 = 0.067$.

Furthermore, nodes can use a threshold value for each edge of the trust chain (β value). If at least one edge of the trust chain has a trust value below this threshold, the chain is discarded. After calculating the trust value for all chains, the trust value $\text{TV}_{(N_x,N_u)}$ can be calculated applying a weighted mean as follows:

$$\text{TV}_{(N_x,N_u)} = \frac{\sum_{i=1}^{k} \left(\text{TC}^i_{(N_x,N_u)} \times \left(1/\left|\text{TC}^i_{(N_x,N_u)}\right|\right)\right)}{\sum_{i=1}^{k} \left(1/\left|\text{TC}^i_{(N_x,N_u)}\right|\right)}. \quad (3)$$

The weighted mean reduces the impact of transitivity in trust chains. In fact, the greater the chain, the less reliable it is. Thus, this method aims to privilege small chains, following a social perspective.

4.4. Key Management. SEMAN employs identity-based cryptography (IBC) [20] to perform its cryptographic operations, which requires an entity acting as a Private Key Generator (PKG). As the PKG knows the master private key, it is able to decrypt or sign messages on behalf of any client, without any active attack and without being detected. This problem is known as key escrow. These issues have been considered the main drawback that leads to the low adoption of IBC outside closed environments [21]. Boneh and Franklin suggested distributing the PKG to handle these problems using secret sharing schemes (n, t), in which n nodes form a distributed PKG (D-PKG) and only a subset of $t + 1$ nodes is able to compute the master private key [11].

Several identity-based key management schemes have been designed for MANETs and most of them consider the distribution of the PKG [9]. However, proposed schemes do not consider all characteristics of these networks.

In a previous work [22], the Identity-based Fully Self Organized (iFUSO) key management service was presented, which can be integrated with SEMAN. The iFUSO considers an asynchronous network composed of n nodes, represented by $N_1, N_2, \ldots, N_n$, in which malicious nodes can compromise at most t nodes and $t < n$. It considers that only trusted nodes participate in the system initialization. Nodes which initialize the system are called founding nodes, denoted by $N_{\mathcal{F}}$.

These nodes form the distributed PKG (D-PKG) in a fully distributed way. No node knows the master private key, since it is distributed in t-over-n threshold scheme. Also, to adapt to the dynamism of the network, *i*FUSO allows nodes to join or leave the D-PKG.

To prevent the system from cryptanalysis attacks, the *i*FUSO provides a way to update public and private keys of nodes, similar to [23, 24]. The key update may occur either periodically according to a predetermined interval, or reactively when the number of revoked nodes reaches a threshold value. Nodes are able to update their public keys autonomously and their private key by requesting to the D-PKG. Further, the *i*FUSO supports both implicit and explicit revocations.

4.4.1. Initialization. The *i*FUSO must be initialized by a set of m founding nodes ($N_{\mathscr{F}}$), which must be able to securely exchange information to initialize the system. As the first step of the initialization, nodes must determine

(1) the system size n and the security threshold t;

(2) p, q: two large prime numbers, in which q divides $(p-1)$;

(3) $\mathbb{G}_1$: a cyclic additive group with order p;

(4) a generator $g \in \mathbb{G}_1$;

(5) $\mathbb{G}_2$: a cyclic multiplicative group with order p;

(6) the pairing type making sure that there exists a bilinear paring $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ of the chosen type;

(7) $\mathscr{G} = \langle e, \mathbb{G}_1, \mathbb{G}_2 \rangle$.

This step can be performed jointly by nodes which are initializing the group, or proposed by one node to the others.

To initialize the system, founding nodes set up the D-PKG, generating the master public key and its corresponding master private key. The D-PKG is built in a distribute t-over-n way among the n founding nodes.

4.4.2. New Members Joining the D-PKG. In a MANET it is very important for a D-PKG to be highly dynamic and decentralized, and new nodes must be able to join the distributed PKG at any time. Thus, these nodes must receive a share of MSK, computed by at least t members of the D-PKG.

If a new node N_k wants to join the D-PKG, it must contact at least t members of the D-PKG to get the corresponding information from these nodes. The join of new nodes to the D-PKG must follow the following:

(1) node N_k selects t members of D-PKG, denoted by Ω;

(2) node N_k requests each node of Ω to be accepted as a member of the D-PKG;

(3) each node $N_j \in \Omega$ sends a piece of information $f(j,k) = S_j^k$ back to N_k;

(4) after receiving t replies, N_k can compute its polynomial share MSK_k using a Lagrange interpolation.

4.4.3. Issuing Node Private Key. The *i*FUSO is composed of a number of continuous, nonoverlapping key update phases, denoted by p_Δ, in which Δ represents the phase index. As in [23], each p_Δ is associated with a unique binary phase string denoted as str_Δ. For each N_i, its public key is represented by $\mathrm{PK}_i = H_1(N_i)$ while the corresponding private key is represented by $\mathrm{SK}_i = (\mathrm{PK}_i)^{\mathrm{MSK}}$. Recalling that in the *i*FUSO no node knows MSK, which is generated and stored in a fully distributed way. To retrieve its private key SK_i, node N_i must request the D-PKG for it and to wait for at least t replies. Thus, the following steps must be executed:

(1) N_i must select at least t nodes from the D-PKG. This set of nodes is denoted by Ψ. To minimize the requesting time, the Ψ has all nodes of the D-PKG;

(2) N_i requests its share of SK_i to each node in Ψ;

(3) each node $N_j \in \Psi$ sends back to N_i a share of the private key $\sigma_i^j = (\mathrm{PK}_i)^{\mathrm{MSK}_j}$;

(4) upon receiving at least t replies, N_i computes its private key SK_i as

$$\mathrm{SK}_i = \prod_{k \in \Psi} \left(\sigma_i^k\right)^{\lambda_k}, \tag{4}$$

in which $\lambda_k = \prod_{k \in \Psi}(k/(k-i))$ are appropriate Lagrange coefficients.

4.4.4. Key Renewing. To prevent attacks against the distributed PKG and potential threats resulting from compromised keys, a technique similar to the one proposed in [23], known as key update or key renewing, is employed. Security solutions based on key update are common practices on MANETs [25–27]. In the *i*FUSO, a new key update phase p_{i+1} starts after a predetermined time threshold. As all nodes must update their keys, if the members of the distributed PKG do not update the key of a given node N_a, it is (implicitly) considered revoked.

Each node N_a can autonomously update its public key $\mathrm{PK}_{a,p_i} = (H_1(N_a), H_1(\mathrm{str}_i))$, in which $\mathrm{str}_i = \mathrm{str}_{i-1} + 1$. On the other hand, generating the phase private key involves at least t members of the D-PKG. A given node N_z, member of the D-PKG, sends a request to at least $t-1$ other members of the D-PKG.

4.4.5. Key Revocation. The *i*FUSO also provides techniques to verify if the public key of a node is revoked. Key revocations must be handled within the system, as nodes must be able to immediately verify the status of a public key [28]. Almost all key management schemes for MANETs consider the key revocation based on expiration time [29]. However, this is not sufficient since nodes must be able to revoke keys before expiration, as consequence of a compromise key or malicious behavior.

Thus, *i*FUSO supports both implicit and explicit revocations. If any node is not allowed to recover the common private key during a given phase p_i, then it is neither able to encrypt nor able to decrypt any information during this phase and is considered implicitly revoked.

On the other hand, the explicit revocation of the *i*FUSO is based on a list of revoked nodes stored by nodes themselves. Upon detecting the misbehavior of node N_a, node N_b generates a signed accusation message against N_a, which must be sent to the D-PKG. To avoid the interception of the accusation message, it is sent via broadcast encryption. This technique, besides decreasing the communication cost of revocation, increases the security since the malicious nodes are not able to read the accusation message.

Upon receiving an accusation message against N_a from N_b, a member of the D-PKG, will drop it if N_b has been previously revoked. Otherwise, it saves the accusation message. To prevent false accusations against legitimate nodes, a node N_a is diagnosed as compromised just when the accusations against it reach a revocation threshold γ in a predetermined time window. The value of γ defines the trade-off between the false accusations tolerance and the compromise detectability. Once the revocation threshold is reached, a key revocation against node N_a is generated and published.

4.5. Group Management. This section presents how the group management must be carried out to support the activities of SEMAN. It this paper, a group is a set of nodes sharing common interests and willing to cooperate on activities related to this interest. This "common interest" is also called *context*. Context information must be frequently updated and made available to other nodes, to allow the efficient organization of groups [30].

Due to several varieties of services provided by a middleware, many kind of distinct groups may be formed, with different mobility pattern, lifetime, organization strategies, internal politics, and joining rules characteristics. However, independent of group characteristics, system must provide resources management to allow the creation and update of existing groups and their profiles.

To support several kinds of applications, with strong and light security restrictions, two group management techniques are used: *yellow pages* and *closed groups*. Yellow pages groups, or open groups, provide primitives allowing nodes to freely form groups and to avail services related to its context. As these groups are open, they do not consider the trustworthiness of their members. Thus, these groups are indicated to services that require a lower trust level or when applications themselves are responsible for this task.

Closed groups use trust management information at their formation. Thus, all services provided by members of a closed context group obey the preestablished security requirements. Also, both internal and external group secure communication are allowed.

4.5.1. Storage Information about Existing Groups. In SEMAN, context groups are considered services available in the network. Information about the existence of groups and their main features must be available to nodes that want to participate in these groups or to enjoy services provided by them. To this end, it is important that the middleware provides ways to manage this information and make it available during its operations. Many architectures were proposed to organize the service provisioning on MANETs. An initial study about these architectures can be found in [31]. Solutions can be classified into with directories and without directories.

In the first approach, information about groups is stored in a directory which can be centralized or distributed. Nodes which store information about directories are called server nodes. Every time a node wants to provide a data, it finds out some server node and requests the storage of this data. On the other hand, a node that wants to use this data needs only to contact a server node and gets a list of nodes which are providing it.

In the second approach, information about groups is not stored in a directory and must be propagated or requested when needed. Thus, when a node wants to provide a service in the network, it diffuses this information, in order to reach the higher number of nodes. It can be employed by global or controlled flooding techniques. When a node wants to use a service and it does not have local information about this service, it requests such an information in the network, via global or controlled flooding.

There is not a consensus about which strategy is more suitable for MANETs. Group discovery is considered good when presenting a high availability, keeping a low communication cost and small delays. Thus, if the network has just few service requirements, a strategy without directories with on demand queries would be more suitable. On the other hand, a network with many services but with few queries would generate unnecessary communications to keep information about these services.

Any one of these architectures may be used on SEMAN. In this work, the use of a fully distributed directories architecture is considered. When a new group is created, its information is disseminated in the network. All nodes locally store information about groups. Thus, every time a node needs information about a group, it may get it locally, without delays or additional costs.

4.5.2. Yellow Pages. The first group formation strategy in the SEMAN is called *yellow pages*. This technique, based in the traditional yellow pages, works as a directory of services, fully open and dynamic. Group formation is directly related to some kind of provided service. When a node wants to provide a service, it informs the middleware, which propagates this information to all other nodes, to make them aware of the new service.

When another node wants to use a service, it requests to the middleware a list of nodes which are providing the service. Based on this list, the node may request or not the service considering information from the trust management module.

This kind of approach is important in providing services without a high level of security. Any node can freely participate in a group and provide a given service. Client applications can, therefore, determinate the trust level they want in a service. Thus, the middleware, based on information provided by the trust management module can select the more trustworthy nodes which are providing the service.

Formation of an Open Group. Before initiating a group, a node needs to certify that there is no other group with the same

characteristics of the new one. To this, it queries its local directory. If the group already exists, the node joins this group (Section 4.5.2). Otherwise, it creates the new group.

When a node wants to form a new context group, it defines all main features of this group, as identifier, mobility pattern, context information, service provided, and initial nodes. Other information may be included to make the group management easier. Then, it disseminates this information through the network, as discussed in Section 4.5.1.

Joining and Leaving an Open Group. When a node wants to participate in an open group G_α, it needs to create a message informing that it is providing the same services as described in the group G_α profile. Then, it must disseminate this information through the network, in order that all nodes be aware that it is providing such services. Note that there is no strategy to block the participation of nodes on open groups. Any node may send a message informing that it is participating in this group.

Similarly, when a node wants to leave a group, it just creates a message informing that it is leaving the group and disseminates this information through the network. However, as this message is not mandatory, or nodes may leave unpredictably and involuntary the network, is necessary some technique are necessary to ensure the consistency of nodes participating in a group. Thus, the node which has created the group, or the older one, periodically performs queries to members, checking their availability. So, at the end of a cycle, a list of available nodes is disseminated into network. It is important that checking interval be not so small in order to prevent a high communication overhead.

Using Secure Services of Open Groups. Open groups do not provide native secure communication methods between members or to service requests. As nodes are able to freely join groups, the group key establishment is difficult. But this does not block that services provided by open groups require ciphered and signed message requests.

When a node wants to request any service to members of an open group, it makes a request directly to these nodes, by using either unicast or multicast messages. If it wants to use ciphered messages, it may use the security primitives provided by the middleware for communication between nodes, supported by cryptographic operations and key management components.

4.5.3. Closed Groups. While some services may be supported by an open group management scheme, other services require a more controlled management. In this case, the middleware provides a dynamic closed group management service to applications. These groups are formed based on applications context, interest, and security requirements.

An example of closed groups is the key management described in Section 4.4 and in [22], which requires a high trustworthy and restrict service. This section describes the closed group management operations, as group formation, new members joining, and leaving.

Closed Group Formation. As previously assumed, groups are formed based on applications context. Each node is able to autonomously promote the formation of a group without a central entity or a group manager. During the group formation, the creator node must only specify the group profile and security requirements.

A group may be composed of a set of nodes, with a unique assumption: these nodes must be able to securely exchange information to initialize the group. Then, to start a group nodes must determinate

(1) group size n and the security threshold t;

(2) p and q: two large prime numbers, in which q divides $(p-1)$;

(3) $\mathbb{G}_1$: a cyclic additive group of order p;

(4) generator $g \in \mathbb{G}_1$;

(5) $\mathbb{G}_2$: a cyclic multiplicative group of order p;

(6) the paring type to ensure that exists a bilinear paring $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ to the choose paring;

(7) $\mathcal{G} = \langle e, \mathbb{G}_1, \mathbb{G}_2 \rangle$.

These values must be defined by all nodes through an agreement approach or may be proposed by a founder node to the other ones. After that, each founder node must have the following public elements:

(1) prime numbers p and q;

(2) generator g and cyclic additive group $\mathbb{G}_1$;

(3) cyclic multiplicative group $\mathbb{G}_2$;

(4) $\mathbb{Z}_q^*$: an elliptic field with order q;

(5) $H_1(x)$: a *hash* function in which $H_1(x) = \{0,1\}^* \to \mathbb{G}_1^*$;

(6) $H_2(x)$: a *hash* function in which $H_2(x) = \mathbb{G}_2 \to \mathbb{Z}_p^*$.

To initialize the group, nodes must generate a public identification of this group and a signature. This signature is distributed between group members by using a threshold cryptographic scheme (m, t) among the m founder nodes, as follows:

(1) Each node N_i chooses a bivariate symmetric polynomial function $f_i(x, y)$ over $\mathbb{Z}_q^*$ in which the two variables x and y must be at most order t. The polynomial function is described as

$$f_i(x, y) = \sum_{k=0}^{t} \sum_{j=0}^{t} a_{k,j}^i x^k y^j, \tag{5}$$

in which $a_{k,j}^i \in \mathbb{Z}_q^*$, $a_{k,j}^i = a_{j,k}^i$, and $a_{0,0}^i = z_i$.

(2) Each N_i computes $f_l^i(x)$ for all N_l founder nodes as

$$f_l^i(x) = f_i(x, l) = \sum_{k=0}^{t} \sum_{j=0}^{t} a_{k,j}^i x^k l^j. \tag{6}$$

Then, N_i securely sends $f_l^i(x)$ to N_l.

(3) Each node N_i computes its share of the signature Sign_i:

$$\text{Sign}_i = f_i(x) = \sum_{j=1}^{n} f_i^j(x) = \sum_{j=1}^{n} f_j(x,i) = f(x,i). \quad (7)$$

(4) Signature Sign is not known by any node but is defined as

$$\text{Sign} = \sum_{N_i \in m} \text{Sign}_i \bmod q. \quad (8)$$

Each node N_i, after computing its share Sign_i, publishes g^{Sign_i}. When nodes receive t shares, they are able to compute the group identity as $\text{ID} = \sum_{i=1}^{t} g^{\text{Sign}_i}$. Then, the group identity ID can be published to all other nodes.

After group formation, nodes which want to collaborate in a specific context or interest must search a group and request its participation in this group. As each group is set with its profile and security requirements, nodes themselves may decide whether available groups attend their own interests.

Joining. As describe in Section 4.5.3, each closed group has its profile and requirements. Thus, nodes themselves may decide on participating or not of a group. If a node N_x wants to join a group G_α it must request to G_α members the participation authorization. To be able to join G_α, N_x needs the approval of at least t members. The following steps must be performed:

(1) node N_x chooses t nodes of group G_α, denoted by Ω;

(2) node N_x requests each node of Ω to be accepted as member of group G_α;

(3) each node $N_j \in \Omega$ sends an information share $f(j,k)$ back to node N_k;

(4) upon receiving t replies, N_x may compute its polynomial share Sign_x by using Lagrange interpolation:

$$\text{Sign}_k = S_k(x) = \sum_{j=1}^{t} \lambda_j S_j^k = \sum_{j=1}^{t} \lambda_j f(j,k) = f(x,k). \quad (9)$$

After computing Sign_x, N_x is able to participate in all group operations.

Members Exclusion. When a node does not attend anymore the security or trust requirement of a group, it must have its participation revoked. To this, signed accusation messages are employed with a list of revoked associations. When a given node N_x has a number of accusations higher than a threshold γ, it has its association revoked. The value of γ is a group parameter, defined on its creation profile.

When a node N_a, based on information provided by trust management, believes that N_x does not satisfy anymore the group requirements, it issues a signed accusation message and sends it to all other group members. To thwart excluded nodes from receiving this message, the signed accusation message must be unknown by them. Thus, a variant of the identity-based broadcast encryption proposed on [32] is used.

Let $\mathscr{E}$ be the set of excluded nodes; then node N_a generates the parameters for the broadcast encryption:

(1) $\forall i \in \mathscr{N} \setminus \mathscr{E}$ computes $\text{PK}_i = H_1(N_i)$;

(2) it randomly selects $r \in \mathbb{Z}_p^*$ and $\forall i \in \mathscr{N} \setminus \mathscr{E}$ computes $s_i = H_2(\hat{e}(\text{PK}_i^r, \text{MPK}))$;

(3) it randomly selects $k \in \mathbb{Z}_p^*$ and computes a message encrypting key $K = \hat{e}(g,g)^k$;

(4) it randomly selects $\alpha \in \mathbb{Z}_p^*$;

(5) it computes $\text{Hdr} = (C_1, C_2, C_3)$ in which

$$C_1 = g^r;$$
$$C_2 = (g^\alpha)^k; \quad (10)$$
$$C_3 = \left\{ c_i = \left(g^{1-1/\alpha}\right)^{1/s_i} \right\}_{N_i \in \mathscr{N} \setminus \mathscr{E}}.$$

After that, N_a has the key K and Hdr and uses K to encrypt the accusation message K, generating C_K. Finally, N_z broadcasts the ciphered message (Hdr, C_K).

When a nonrevoked node N_b receives this message, it is able to retrieve the accusation message K encapsulated in the header Hdr, using its private key SK_b, as follows:

(1) computes $s_i = H_2(\hat{e}(\text{SK}_b, C_1))$;

(2) retrieves c_i from C_3 and computes

$$\begin{aligned} &\hat{e}\left(C_2^{-1}, c_i^{s_i}\right) \times \hat{e}(g, C_2) \\ &= \hat{e}\left(\left((g^\alpha)^k\right)^{-1}, \left(\left(g^{1-1/\alpha}\right)^{1/s_i}\right)^{s_i}\right) \times \hat{e}g, (g^\alpha)^k \\ &= \hat{e}(g,g)^{-k(\alpha-1)} \times \hat{e}(g,g)^{k\alpha} = K. \end{aligned} \quad (11)$$

With K, node N_b is able to decrypt the encrypted message C_K, extracting the accusation message K. Upon receiving γ accusations, each node creates an association revocation register and stores it locally in a revoked associations list. This list may be made publicly available to all nodes, in order that external member be aware that N_x is no longer authorized to provide service in name of the group.

4.6. Conclusion. This section presented the security module and its components. Cryptographic operations, trust management, group management, and key management components were detailed. Also, how these components will ensure security to the middleware was explained.

5. Secure Group Communication

To allow secure group communication, the use of a group key agreement protocol is proposed. This kind of protocol allows that a group of users exchange information over an insecure and public communication channel and agree on a secret key to be used to derive a session key. Then, the session

key can be used to ensure requirements as authentication, confidentiality, and integrity.

The group key agreement approach is attractive to dynamic networks since it does not require the presence of a central controller or a leader. In this case, all users generate the key session. Thus, no node is able to control or to predict the session key. This kind of approach has been widely employed in distributed and collaborative applications, as file sharing, distributed computing, and audio and video conferences, among others.

Several proposals to session group key establishment can be found in the literature [33–35]. Any scheme that makes use of identity-based approach may be easily employed in the SEMAN. Without the loss of generality, the scheme proposed by Zhang et al. in [35] is assumed. An advantage of this scheme is that it allows outside group nodes to send ciphered messages to group members. This makes the secure service request to closed groups easier.

In the key agreement, members of a group G_α issue signed messages. The union of all signed messages issued by them form a group encryption key, called GEK_α, which may be public. However, only members of the group are able to derive the group decryption key GDK_α. The next subsections present the key agreement operations and the group encryption and decryption key generation.

5.1. Agreement. A given node N_i, with the private key SK_i and member of group G_α, must perform the following steps to carry out the key agreement:

(1) to choose a random number $\eta_i \in \mathbb{Z}_q^*$;

(2) to compute $r_i = g^{\eta_i}$;

(3) to choose a random number $k \in \mathbb{Z}_q^*$;

(4) to compute $g_1 = g^k$;

(5) for all $1 \le j \le n$, to compute $f_j = H_2(N_j)$, in which $H_2(x)$ is a hash function;

(6) for all $1 \le j \le n$, to compute $z_{i,j} = \mathrm{SK}_i f_j^{\eta_i}$;

(7) to publish $\sigma_i = (r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\ldots,n\}, j\neq i})$.

In this case, ϱ_i is the identity-based signature on value r_i. Element $z_{i,j} = \mathrm{SK}_i f_j^{\eta_i}$ is not published, but kept secret by node n_i.

5.2. Encryption Key Generation and Use. To get a group encryption key, a node firstly checks the n tuple of signature message $(r_1, \varrho_1), \ldots, (r_n, \varrho_n)$. If all signatures are valid, then it computes

$$w = \prod_{i=1}^{n} r_i,$$
$$Q = \widehat{e}\left(\prod_{i=1}^{n} H_1(N_i), g_1\right). \tag{12}$$

Then, it sets the group encryption key as $\mathrm{GEK} = (w, Q)$. To encrypt a message M, any node, member or not of the group, generates a ciphered text as follows:

(1) selects $\rho \in \mathbb{Z}_q^*$;

(2) computes $c_1 = g^\rho$, $c_2 = w^\rho$, and $c_3 = M \oplus H_3(Q^\rho)$;

(3) generates the ciphered text $c = (c_1, c_2, c_3)$.

After the ciphered text c generation, it can be sent through the network and only members of destination group are able to decrypt the transmitted message.

5.3. Decryption Key Generation and Use. Each node N_i checks the n tuples of signed messages $(r_1, \varrho_1), \ldots, (r_n, \varrho_n)$. If all signatures are valid, node N_i computes $\mathrm{GDK} = \prod_{j=1}^{n} z_{j,i}$ and makes the following verification:

$$\widehat{e}(\mathrm{GDK}_i, g) \stackrel{?}{=} \widehat{e}(f_i, w) \cdot Q. \tag{13}$$

If the equation is correct, node N_i accepts the key GDK as decryption group key. Otherwise, it aborts the procedure.

When a node N_i, member of group, receives a ciphered message $c = (c_1, c_2, c_3)$, it uses the decryption key GDK as

$$M = c_3 \oplus H_3\left(\widehat{e}(\mathrm{GDK}, c_1)\,\widehat{e}\left(f_i^{-1}, c_2\right)\right). \tag{14}$$

5.4. Conclusion. This section presented how to provide a secure group communication through SEMAN. How nodes may use cryptographic services to make an agreement and provide secure group communication to applications was discussed.

6. Components Integration in Different Scenarios

Policy management component is responsible for supporting the security module integrating the trust, key, and group management components. Thus, the development of strategies is fundamental to provide security in several scenarios in which applications may be provided by SEMAN.

This section discusses some cases, which show how the middleware may be used in different scenarios. Security parameters described to each scenario are configured on the security policy component, part of the security module. It is important to point out that several scenarios can be found in a unique network. Some applications may be better suitable to open scenarios, while other ones require a more rigorous security control. SEMAN allows the configuration of these different scenarios, since applications and nodes are organized into context, with profiles and security parameter set according to provided services.

Three scenarios are presented:

(i) *open*: indicated to applications which do not require a high security control;

(ii) *partially restrict*: indicated to applications which require an intermediary security support but do not require a rigid control on their operations;

(iii) *restrict*: indicated to applications which require a high security level on their operations.

To each presented scenario, distinct security policies are indicated. Table 1 illustrates how security components

Table 1: Security policies for distinct scenarios.

	Trust	Key	Groups
Restrict	Block evidence exchanges with nonreliable nodes	Allow the service to be provided to each distinct group t value greater than $n/2$	Prioritize closed group creation Trust restriction to group joining
Partially restrict	Trust values of α and β between 0.4 and 0.6	A unique system to all the middleware t value greater than $n/2$	Prioritize closed group creation Allow the service providing in closed groups
Open	Trust values of α and β less than 0.4	A unique system to all the middleware Small t value Great interval between updates	Most part of provided services in open groups

may be set to proposed scenarios. However, these scenarios parameters and politics are not static, and new environments and configurations may be proposed and configured by middleware users.

Next subsections detail these scenarios and how security components may be integrated in the service provisioning to applications. Each scenario discusses how SEMAN can ensure the desired security and the communication overhead imposed by it. However, the measurement of this kind of overhead is a complex task, since these values depend on several factors: group size, update interval, threshold values, and others. Then, a more realist approximation of the overhead is a future work.

6.1. Open Scenarios. A first scenario that SEMAN may be employed is an open environment, which requires a less rigorous security control. Several applications may provide services in an open scenario. An example is the data and file distributed storage service. In this case, users may share, for a while, data or files that do not require a high level of confidentiality and availability, as video or audio files.

(1) *Trust management*: adopting TRUE, values of α and β may be small, less than 0.4. Then, middleware will consider more nodes reliable on evaluated context. As a consequence, more nodes will be considered reliable to provide services in this context.

(2) *Key management*: using *i*FUSO, values of t may also be small, less than $m/2$, in which m is the members of D-PKG. Also, update interval may be high. Then, system overhead is reduced while the service is offered to users considering the defined parameters.

(3) *Group management*: in open scenarios, services may be provided in open groups, and users themselves can query the trust management component to decide by the use or not of the offered services.

Note that, in this scenario, a unique great group may be used to the key management of the entire middleware. Then, all applications which need cryptographic services use the same service.

To all other services provided in the network, the trust management may provide trust information on the context of these services. For example, a resource localization service have a different context than a distributed storage. Nodes which provide these services may be organized into open groups, but their users may use trust values to choose the best servers to their requirements.

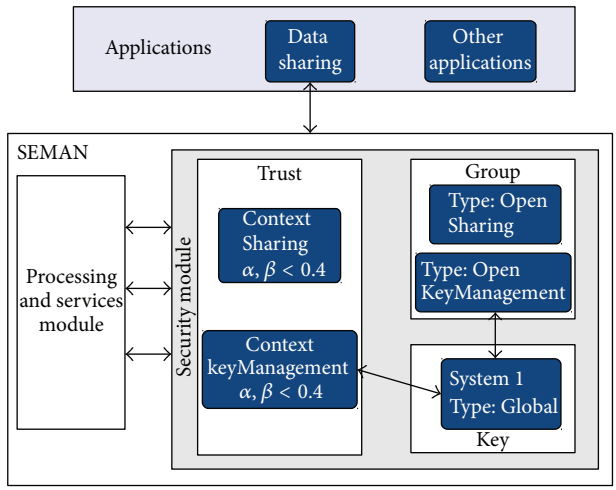

Figure 9: Open scenario.

Figure 9 illustrates a possible configuration of the security module to satisfy applications in an open scenario. In this case, the "data sharing" application requests services to the middleware. Services and processing modules reply application requests and, if necessary, make queries to the security module. In this scenario, trust management provide two contexts: "sharing" and "KeyManagement" Both have α and β values lower than 0.4.

The context KeyManagement is queried by the key management system for key issuing, revocation, and update. Though trust management values will be considered in the key issuing, threshold values to the acceptance are small. Thus, any node may request participation as a D-PKG member, making the group open, but the private master key is issued only if this node satisfies a minimal trust requirements.

The other context is called Sharing and may be queried by applications on the acceptance or not of services provided by members of an open group named "Sharing." Note that any node is able to participate in this group. If necessary, group members or client applications may query the global key management system to check the identity authenticity of any group member.

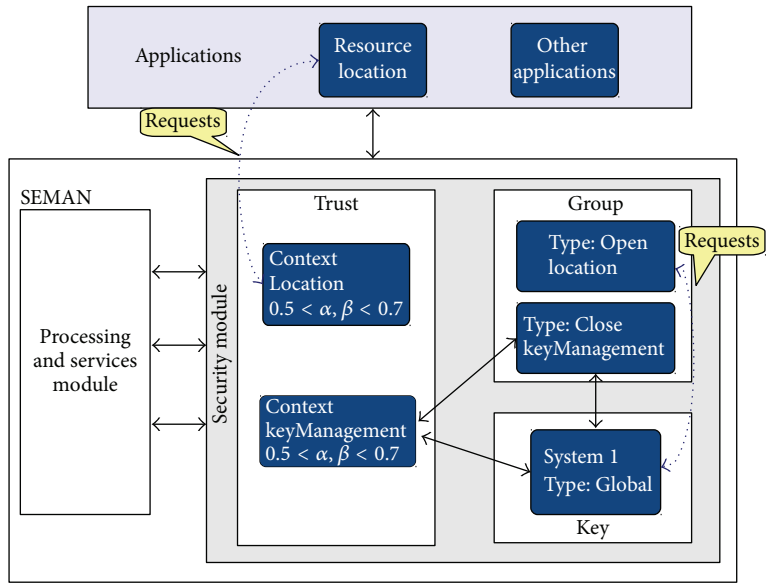

Figure 10: Partially restrict scenarios.

With these configurations, middleware provides just some security guarantees to applications. Applications which perform queries to trust management may receive unreliable information, since system is susceptible to false accusation attacks. The key management is vulnerable to impersonation attacks, since an attacker needs to compromise a small number of D-PKG members to be accepted and to receive its own private key or even to be a D-PKG member.

6.2. Partially Restrict Scenarios. A second scenario is a partially restrict scenario, in which an intermediary security control is necessary. A service example which can be classified as partially restrict is the resource localization. In this case, it is important to provide guarantees of nodes authenticity, but at same time it does not need to block any node to provide a service to applications.

In this case, security parameters and threshold values may be set with more restrictions than previous scenario. Hereafter, some suggestions are presented:

(1) *trust management*: using TRUE, α and β values may be between 0.5 and 0.7. Then, middleware will exchange information with nodes that have an intermediary trust evaluation in the context. As a consequence, the communication overhead will be smaller than previous scenario, while it keeps a higher control of transmitted information;

(2) *key management*: with *i*FUSO, t value for the master key sharing should be higher than $m/2$, to prevent network partitioning attacks. Further, key update intervals may not be large, to block unreliable nodes to be on the system for a long time;

(3) *group management*: in this scenario, some applications may be provided in closed groups, but the majority may be organized into open groups. Thus, even all nodes being able to join a group and to provide services in this context, client applications may use trust management information to select the best node to request a service.

Note that in this scenario, a unique group may be used to the middleware-wide key management. Thus, all applications which need cryptography use the same service. As on open scenarios, the trust management must have information of a context which will be queried by key management, for example key-management. For all other provided services, the trust management may provide trust information on their own context.

Figure 10 illustrates how SEMAN components may be integrated to satisfy security requirements on partially restrict scenarios. "Resource location" application requests services to the middleware, which are received by the services and processing modules. When necessary, some queries are made to the security module. Two contexts are envisaged: KeyManagement and Location. On both contexts, trust management α and β values are between 0.5 and 0.7.

As in open scenarios, KeyManagement context is queried by the key management for key issuing, revocation, and update. However, only nodes of the closed group named "KeyManagement" are accepted to be members of the D-PKG. Thus, the participation of nodes as D-PKG members is more restrictive, making the system more reliable. Also, α and β parameters of the KeyManagement context may be increased, to ensure more security to key management and to client applications.

The other context is called Location and may be queried by application itself in the acceptance or not of services provided by members of group named "Location." As in the open groups, any node may be member of this group and, if necessary, client applications may query the global key management to check the identity authenticity of a node.

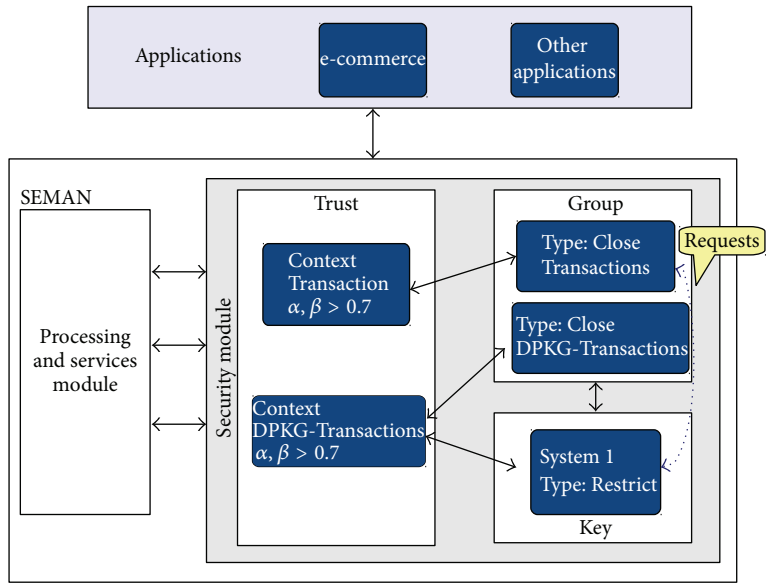

Figure 11: Restrict scenarios.

With these configurations the system should be protected against impersonation attacks. Also, higher α and β values make the propagation of false accusations against a node reputation more difficult. Thus, the middleware ensures, to users, the authenticity of nodes which are participating in these groups.

6.3. Restrict Scenarios. The third scenario is a restrict environment, in which a more rigorous security control is necessary. This context considers the applications which cannot be compromised in face of attacks. These applications perform, in general, essential tasks to users and, if affected by malicious attacks, may compromise the integrity of provided services. Examples are the e-commerce or financial transactions. These services may receive the guarantee from the middleware that they are protected against malicious attacks.

In this case, security parameters and threshold values must be set with many restrictions. The following are some suggestions:

(1) *trust management*: if TRUE is adopted, α and β values must be higher than 0.7. Then, middleware will exchange information only with reliable nodes in related context. As a consequence, less nodes will be able to join groups of this context;

(2) *key management*: with *i*FUSO, t values for master key sharing must be higher than $m/2$. Further, update phase interval cannot be high. Then, the system overhead is increased, but services provided will ensure security against malicious attacks;

(3) *group management*: in restrict scenarios, applications must be provided in closed groups, with a more restrictive control on group joining.

Note that for this kind of scenario, different key management group may be used, for each application. Thus, more restrictive applications may have their own context-related key management. This blocks malicious nodes, even the ones not participating in the group that is providing a service, to be a D-PKG member. On the other hand, a global key management scheme may be implemented to provide service to groups with less restrictive applications. For all other services, the trust management may provided trust information on the services context.

Figure 11 illustrates the integration of SEMAN components on restrict scenarios. E-commerce application requests services to middleware. Requests are received by the services and processing modules that if necessary query the security module. Two contexts are envisaged: DPKG-Transaction and Transaction. On both contexts, trust management has α and β values higher than 0.7.

The DPKG-Transaction is queried by the key management of closed group "Transaction." To this, group management allows the formation of a closed group called "DPKG-Transaction," which has only the nodes that satisfy the requirements to be a D-PKG member. These D-PKG members make queries to the closed group "Transactions" to check nodes reliability and issue private keys to them.

The other context is called Transactions and it is queried by members of closed group with same name, which are responsible for the acceptance or not of a new member, or by the exclusion of a current member. Unlike previous scenarios, to participate in a group, nodes need to satisfy the trust requirements defined by group policies, increasing the system security and protection against malicious attacks.

6.4. Hybrid Scenarios. Three distinct scenarios of security module configuration were presented. However, in practice, each application which is using the middleware services may present a different scenario. For example, at the same time, middleware may provide services to applications that require a high level of security and also to an open one. Thus, SEMAN

security policies must be directed to applications and services contexts which use the security module.

A recommendation to the use of SEMAN in these scenarios is the setting of a global key management, which satisfies all services. Thus, the entire network is supported by a unique D-PKG and all users have a unique public/private key pair to use on all applications. Trust management must offer information about users in two contexts: key-management and key-management-dpkg.

Trust information about first context (key-management) is used by key management when D-PKG members perform users private key issuing, update, or revocation. In this case, users with trust value in this context less than a threshold (that must be high) will not have their keys issued or update by D-PKG members.

Trust information of the second context (key-management-dpkg) is used by D-PKG members on decisions about the acceptance or not of new D-PKG member. These restrictions must be higher than the key issuing restrictions. For example, a node can be authentic and has the right to have its private key issued by a D-PKG. However, it cannot be reliable enough to be member of D-PKG.

With a secure key management, middleware ensures the protection of cryptographic operations against malicious attacks and the authenticity of members. Thus, each application may be provided inside a context of closed or open groups, which requires a high level of reliability or allows that services to be provided by any node. Thus, each application will be protected against malicious actions depending of its own policies.

6.5. Conclusion. This section presented a study of some scenarios which SEMAN may be employed to satisfy applications requirements.

7. Conclusions and Future Directions

This paper proposed secure context-based middleware, called SEMAN, which employs a group approach to support security making decisions. The middleware architecture and an overview of its operations were discussed. Also, how SEMAN services must be provided and how group approach may be applied to ensure the security were presented.

SEMAN has three modules: services, processing, and security. The first two are responsible for providing services and for requests management. Security module is responsible for ensuring the security to applications that use the services provided by SEMAN. This module is composed of trust, key, and group management components. All these components were detailed, focusing on how they can be used to ensure the security to applications.

Security module components are integrated through policies management which is responsible for security parameters of each provided service. Further, all activities are supported by identity-based cryptographic operations. The integration of these components was discussed in distinct scenarios, in which some configuration suggestions were presented to satisfy applications requirements.

Table 2: Expected communication overhead.

	Trust	Key	Groups
Restrict	Low	Low	High
Partially restrict	Medium	Low	Medium
Open	High	Very low	Low

SEMAN provides security against selfish, impersonation, Sybil, and byzantine attacks. But other attacks can be found on MANET and may affect the middleware efficacy. At the end, some scenarios in which SEMAN may be employed were presented. These scenarios are classified into open, partially restrict, and restrict. Table 2 illustrates the expected overhead in each presented scenario. SEMAN does not impose a high communication overhead. Key management, for example, needs more messages exchanges during a group creation and update. However, key updates do not occur frequently.

Group management presents an overhead that depends on the way groups are organized. Closed groups have higher overhead than open ones. However, even closed groups have a higher communication cost only during group formation. Further, secure group communication allows multicast messages, decreasing the quantity of individual messages.

Trust management imposes a higher communication overhead when groups are open and, then, α and β values are smaller. However, even this higher quantity of messages is performed just among neighbor nodes, not affecting the entire network. Then, security module of SEMAN can be used to ensure the security requirements of application while not imposing a high communication overhead to network.

To increase the system reliability, new services can be integrated to SEMAN and may be performed in future work, as integrating with analysis tools of external environment, to help automatic and dynamic configuration of security policies; proposing the integration of SEMAN with other certificateless public key cryptography; implementing and evaluating the middleware in real scenarios; and designing an accounting scheme integrated with trust management to impede that denial of service attacks overhead the system with false control messages. Further, a study should be performed in order to reduce de requirements of exponential and pairing computations. Authors suggest the use of certificateless public key cryptography, or other alternative models, comparing their computational cost.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This study is partially funded by CNPq, Grants 448004/2013-6.

References

[1] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks," in *Wireless Network Security*, Y. Xiao, X. S. Shen, and D.-Z. Du, Eds.,

Signals and Communication Technology, chapter 12, pp. 103–135, Springer, New York, NY, USA, 2007.

[2] P. Papadimitratos and Z. J. Haas, "Securing mobile ad hoc networks," in *The Handbook of Ad Hoc Wireless Networks*, chapter 21, pp. 457–481, CRC Press, Boca Raton, Fla, USA, 2005.

[3] R. Shirey, *RFC 2828: Internet Security Glossary*, EUA, Marina del Rey, Calif, USA, 2000, http://www.ietf.org/rfc/rfc2828.txt.

[4] E. da Silva and L. C. P. Albini, "Middleware proposals for mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 43, pp. 103–120, 2014.

[5] P. A. Bernstein, "Middleware: a model for distributed system services," *Communications of the ACM*, vol. 39, no. 2, pp. 86–98, 1996.

[6] J. Al-Jaroodi, I. Jawhar, A. Al-Dhaheri, F. Al-Abdouli, and N. Mohamed, "Security middleware approaches and issues for ubiquitous applications," *Computers and Mathematics with Applications*, vol. 60, no. 2, pp. 187–197, 2010.

[7] S. Hadim, J. Al-Jaroodi, and N. Mohamed, "Trends in middleware for mobile ad hoc networks," *Journal of Communications*, vol. 1, no. 4, pp. 11–21, 2006.

[8] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, 2003.

[9] E. da Silva, A. L. dos Santos, L. C. P. Albini, and M. N. Lima, "Identity-based key management in mobile ad hoc networks: techniques and applications," *IEEE Wireless Communications*, vol. 15, no. 5, pp. 46–52, 2008.

[10] H.-Y. Chien and R.-Y. Lin, "Improved ID-based security framework for ad hoc network," *Ad Hoc Networks*, vol. 6, no. 1, pp. 47–60, 2008.

[11] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology—CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer, London, UK, 2001.

[12] X. Li, J. Slay, and S. Yu, "Evaluating trust in mobile ad hoc networks," in *Proceedings of the Workshop of International Conference on Computational Intelligence and Security (CIS '05)*, Springer, Xi'an, China, 2005.

[13] J. van der Merwe, D. Dawoud, and S. McDonald, "A survey on peer-to-peer key management for mobile ad hoc networks," *ACM Computing Surveys*, vol. 39, no. 1, article 1, Article ID 1216371, 2007.

[14] M. Nogueira, G. Pujolle, E. Silva, A. Santos, and L. Albini, "Survivable keying for wireless ad hoc networks," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM '09)*, pp. 606–613, IEEE Communications Society, Long Island, NY, USA, June 2009.

[15] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of the 17th IEEE Symposium on Security and Privacy (SP '96)*, pp. 164–173, Oakland, Calif, USA, May 1996.

[16] M. Misaghi, E. da Silva, and L. C. P. Albini, "Distributed self-organized trust management for mobile ad hoc networks," in *Networked Digital Technologies*, R. Benlamri, Ed., vol. 293 of *Communications in Computer and Information Science*, pp. 506–518, Springer, 2012.

[17] E. da Silva, M. Misaghi, and C. P. Luiz, "True: a trust evaluation service for mobile ad hoc networks resistant to malicious attacks," *Journal of Digital Information Management*, vol. 10, no. 4, pp. 262–271, 2012.

[18] J. W. Mickens and B. D. Noble, "Modeling epidemic spreading in mobile environments," in *Proceedings of the 4th ACM Workshop on Wireless Security (WiSe '05)*, pp. 77–86, ACM, Cologne, Germany, September 2005.

[19] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Computer Networks*, vol. 51, no. 10, pp. 2867–2891, 2007.

[20] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, New York, NY, USA, 1985.

[21] A. Kate and I. Goldberg, "Distributed private-key generators for identity-based cryptography," in *Security and Cryptography for Networks*, J. A. Garay and R. De Prisco, Eds., vol. 6280 of *Lecture Notes in Computer Science*, pp. 436–453, Springer, Berlin, Germany, 2010.

[22] E. da Silva and L. C. P. Albini, "Towards a fully self-organized identity-based key management system for MANETs," in *Proceedings of the 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '13)*, pp. 717–723, Lyon, France, October 2013.

[23] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 386–399, 2006.

[24] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: ubiquitous and robust access control for mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1049–1063, 2004.

[25] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[26] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *Proceedings of the International Conference on Network Protocols (ICNP '01)*, pp. 251–260, Washington, DC, USA, November 2001.

[27] S. Yi and R. Kravets, "MOCA: mobile certificate authority for wireless ad hoc networks," in *Proceedings of the 2nd Annual PKI Research Workshop (PKI '03)*, National Institute of Standards and Technology (NIST), Gaithersburg, Md, USA, 2003.

[28] K. Hoeper and G. Gong, "Key revocation for identity-based schemes in mobile ad hoc networks," in *Ad-Hoc, Mobile, and Wireless Networks*, T. Kunz and S. S. Ravi, Eds., vol. 4104 of *Lecture Notes in Computer Science*, pp. 224–237, Springer, 2006.

[29] V. Daza, P. Morillo, and C. Ràfols, "On dynamic distribution of private keys over MANETs," *Electronic Notes in Theoretical Computer Science*, vol. 171, no. 1, pp. 33–41, 2007.

[30] O. Courand, O. Droegehorn, K. David et al., "Context aware group management in mobile environments," in *Proceedings of the 14th IST Mobile and Wireless Communications Summit*, Nokia Research Center, Dresden, Germany, 2005.

[31] C. N. Ververidis and G. C. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, 2008.

[32] J. Hur, C. Park, and S. O. Hwang, "Privacy-preserving identity-based broadcast encryption," *Information Fusion*, vol. 13, no. 4, pp. 296–303, 2012.

[33] D. Augot, R. Bhaskar, V. Issarny, and D. Sacchetti, "An efficient group key agreement protocol for ad hoc networks," in *Proceedings of the 6th IEEE International Symposium on a World*

of Wireless Mobile and Multimedia Networks (WoWMoM '05), pp. 576–580, Washington, DC, USA, June 2005.

[34] B. E. Jung, "An efficient group key agreement protocol," *IEEE Communications Letters*, vol. 10, no. 2, pp. 106–107, 2006.

[35] L. Zhang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Provably secure one-round identity-based authenticated asymmetric group key agreement protocol," *Information Sciences*, vol. 181, no. 19, pp. 4318–4329, 2011.