Georg-August-Universität Göttingen
Institut für Informatik

A Bio-Inspired Autonomous Authentication Mechanism in Mobile Ad Hoc Networks

Dissertation
zur Erlangung des
mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

Vorgelegt von

Parisa Memarmoshrefi

aus

Teheran/Iran

Göttingen 2012

Referent:
Prof. Dr. Dieter Hogrefe

Korreferent:
Prof. Dr. Junichi Suzuki
Department of Computer Science
University of Massachusetts, Boston

Mitglieder der Prüfungskommission:
Prof. Dr. Carsten Damm
Prof. Dr. Xiaoming Fu
Prof. Dr. Jens Grabowski
Prof. Dr. Dieter Hogrefe
Prof. Dr. Junichi Suzuki
Prof. Dr. Stephan Waack

Tag der mündlichen Prüfung: 30 Mai 2012

Abstract

A Mobile Ad hoc NETwork (MANET) is an infrastructure-less self-configuring network in that nodes themselves create and manage the network in a self-organized manner. Nodes can communicate only with their neighbors that are located within their wireless range. This dependency on intermediate nodes and lack of an infrastructure cause security challenges in MANETs. Authentication as fundamental security service is required for secure communication. In this work we focus on an autonomous authentication mechanism. A successful authentication mechanism or key management system is highly dependent on the trusted key and trusted key exchange. In this work, we consider asymmetric public-private key pairs scheme as key management technique. Among different proposed authentication mechanisms for MANETs, fully self-organized scheme are more appropriate for these environments. In authentication mechanisms like Public Key Infrastructure (PKI), there is a centralized authority. The central authority is fully trusted by all participants in the network and is responsible for checking the authenticity of the nodes' public keys. Due to the characteristic of MANETs, like lack of infrastructure and frequent topology changes, PKI is infeasible for these networks.

In self-organized public key management, all tasks including key generation, distribution, storage and revocation of keys are performed locally by participants themselves. Therefore existence of attackers who aim at sabotaging the authentication process is unavoidable. In this regard, trust relationship is needed to be established between nodes. Trust-based mechanisms are applied to maintain security by identifying trustworthy and untrustworthy nodes. In our work the scope of trust is identity trust. Identity trust means to assure the identity of a node that it claims to be. Every node generates its own public-private key pair and issues certificates to its neighboring nodes. In order to find the correct public key of a target node for secure communicating, an on-demand authentication service by means of gathering certificate chains towards a target node is initiated. To form the autonomous authentication service, a learning process is needed to distinguish between trustworthy and untrustworthy nodes.

The cooperative and self-organized nature of the MANETs makes ant colony optimization (ACO) suitable for such environments. In the field of telecommunication they are applied for routing tasks. ACO is inspired from distributed and collaborative behavior of real ant colonies in order to construct the shortest path from nest to a

source of food. Volatile chemical substance called pheromone, laid on the ground as the trace of ants and affect their moving decisions. Paths with higher density of pheromone value attract more ants. This in turn increases the pheromone value of these paths. Identifying shortest path by pheromone traces happening over time is considered as a collective learning process.

In our proposed self-organized and localized public key authentication mechanism based on ant colony systems, pheromone concentration left by ants along the path of the certificate chains represents the trust level of a node towards other nodes. As the form of pheromone updating process, our trust updating is aggregated with an incentive mechanism include punishing and rewarding processes. The incentive mechanism is adaptive to the environment with malicious public key certificate signers. It evaluates the certificate chains gathered via a request that source nodes makes to find the public key of a destination. Our certificate chain evaluation process identifies a chain consisting of malicious nodes. This model is able to authenticate public keys by selecting the most trustworthy path in certificate chains gathered by ants and can identify and prevent certificate chains with malicious nodes. Our authentication mechanism is able to retrieve the public key certificate of a destination despite of malicious signers in the network. Our scheme has also the ability to efficiently adapt itself to dynamic environments.

Acknowledgement

I would like to thank my supervisor, Prof. Dieter Hogrefe and express my deep and sincere gratitude to him for his support, guidance and invaluable advices during my work. He created a stable and motivating environment in which I could have the freedom in my work. Besides his great supportive supervising I highly appreciate his kindness and humility.

I also would like to gratitude Prof. Junichi Suzuki for reviewing my dissertation and for the labor to attend my disputation coming from a long distance.
My thanks and respect go to the members of my thesis jury, Prof. Dr. Carsten Damm, Prof. Dr. Xiaoming Fu, Prof. Dr. Jens Grabowski and Prof. Dr. Stephan Waack.

I owe Roman Seibel many thanks for the constructive discussion and valuable collaboration during my research. My gratitude goes to Ansgar Kellner for the helpful scientific discussions and reviews. Also I am thankful for the cooperative efforts of Hang Zhang. I am grateful to all of my colleagues; Youssef for the technical help and to Somayeh and all others for their kind cooperation. I wish them all the best for the future.

It is a pleasure for me to thank Carmen Scherbaum and Udo Burghardt who create a gracious work atmosphere in the Telematics group with fast administrative environment. I am also thankful to Annette Kaziora and Heicke Jachinke for their kind support.

I am very indebted to my beloved family, my parents and my brothers, who have been, from distance, a great support. I would like to gratitude to my parents for their love, support and encouragement in different phases of my life.
The final word of thanks goes to my husband Kamyar. I appreciate his support, patient and encouragement during all steps of my work. This work is dedicated to him and to my parents.

Contents

List of Figures

List of Tables

The broader the mind,
the greater the tolerance .
-Ostad Elahi

Chapter 1 Introduction

A wireless Mobile Ad hoc NETwork (MANET), often termed as self-organized, is built on the cooperation between two or more network nodes. MANETs have a variety of applications where infrastructure-less communication is required. Major applications of such networks include emergency relief, rescue mission, tactical military, on-demand conferencing and in-home patient monitoring systems. Mobile units, e.g., laptops or cars, are equipped with wireless communication devices to form ad hoc networks and roam in insecure environments. The nodes are independent units which do not rely on any central infrastructure. The lack of a pre-existing infrastructure in such networks causes all network functionalities, e.g., routing, security, network management etc., to be performed by the nodes themselves. Consequently, an ad hoc network can work properly only if the participating nodes cooperate with each other and do not disobey the designed rules. These characteristics have a significant impact on the design of security. Due to the lack of any infrastructure, attacks on wireless ad hoc networks can target any node. Therefore providing security is the most critical challenge in MANETs. However, ensuring a secure environment for wireless networks is different from ensuring security for wired ones. In [1] security mechanisms are classified into two classes, called *hard security* and *soft security* approaches.

Hard security has lines of defense, such as firewalls and gateways, which are applied for providing security in wired networks. When these lines are bypassed, secret data will be revealed. *Soft security* approaches emerge to solve the problem. With these mechanisms, as applied in social networks, it is the participants themselves who are responsible for ensuring the different security aspects of an environment. Soft security mechanisms do not prevent the existence of misbehaving nodes; however, with these mechanisms malicious nodes can be identified and isolated from the network’s functionalities. In such social control-based security approaches, the notion of *trust* is of significant importance. It is the belief in reliability, honesty and cooperativeness of an entity.

So far, several trust and reputation systems have been proposed for dealing with malicious behavior in different domains, such as human social networks, e-commerce [2], peer-to-peer networks [3, 4][5], mobile ad hoc networks [6][7] and sensor networks [8, 9].

Due to the infrastructure-less and distributed nature of ad hoc networks, trust-based systems are proposed for these environments. In trust-based systems security is achieved upon access to the history of each participant’s behavior, which is used to calculate the trustworthiness of the nodes. Each node should be capable of making its own security decisions via cooperating with other peer nodes and their corresponding calculated trust values.

In [10] the authors emphasize that trust and security are two interdependent issues. The notation of trust covers a variety of network security requirements. [2] present different trust classes, like *resource access trust, service provision trust* and *identity trust*. *Identity trust*, also called *authentication trust*, is the fundamental type of trust on which the other types are built. This trust domain denotes the confidence which is achieved when an agent's identity is the same as it claims to be.

Among the proposed trust-based systems, most assume the existence of identity trust in which the identity of an entity in the environment is fully trusted. The field of security in ad hoc networks initially focused on service provision trust, such as secure routing protocols [11]. These protocols aimed at providing routing mechanisms that are robust against a dynamic topology and the existence of malicious nodes. However, all routing schemes neglect identity trust and the crucial task of the *authentication* mechanism. In routing schemes, often the pre-existence of an authentication mechanism and pre-sharing of key materials are assumed. Authentication is considered as fundamental part of any secure communication [12]. In order to provide secure communication, data should be encrypted and authenticated. Authentication ensures that the other end of the connection, or the originator of data, is the node it claims to be. Therefore, before a secure communication can be established, the identities of the entities, which are involved in communication, should be confirmed. For establishing identity trust an effective key management system is required.

A classification of authentication mechanisms in MANETs is presented in [13], categorizing three different key management schemes: 1) central certification authority (CA) systems; 2) distributed CA systems; 3) self-organized CA systems. In centralized CA there is a unique authority center, fully trusted by all participants. The central authority is the only entity in the network that is responsible for verifying the authenticity of the participants; however, CA can be the single point of failure and, therefore, it is not suitable for providing secure communication in mobile ad hoc networks. In the distributed CA scheme, the mission of the authentication process is shared between several *n* nodes. Any *k* out of these *n* nodes in the network collectively performs the task of a CA. In order to fulfill the authentication process, simultaneous accessibility to *k* nodes is required; this makes this scheme inappropriate for a mobile environment as in MANETs.

The latter model, the self-organized CA system, allows nodes to become an individual CA and to perform the authentication process. The authentication process is performed autonomously without relying on any predefined certification authority. All nodes in the network have the same level of responsibility. In the self-organized model each node generates its own public-private key pairs. Each node keeps its private key secret and makes its public key available to others.

However, there are security threats in this autonomous and unsupervised ad hoc environment because of the absence of a centralized authority. The main problems of this approach involve (i) protecting public keys from tampering; (ii) making each node's public key available to the other nodes such that its authenticity is verifiable.

A public key certificate is used to prevent a public key from tampering. A node uses a certificate as proof of its identity. A public key certificate is a data structure in which a public key is bound to an identity and signed by the issuer of the certificate. For example, a node's identity, here node *S*, is proven by node *B* as in the following procedure: Node *B* obtains the public key and the identity of node *S*. If node *B* trusts *S*, it composes a certificate that binds *S'* identity and public key. Node *B* signs the certificate with its own private key, verifying that *S'* public key correctly belongs to *S*. Other nodes can therefore obtain this certificate, issued by *B* towards *S'* public key. In the self-organized certification authority scheme there is no fully trusted center to do the task of authentication. Therefore, trust relationships are created by issuing certificates.

As in mobile ad hoc networks, nodes are in contact with their neighbors that are in the radio range of each other; the problem arises when they request a public key of a target node out of their range. This is solved by a chain of certificates, inspired by PGP [14]. With this model a fourth factor, as proposed by [15], come into consideration while authenticating an entity. This factor is referred to as *somebody you know*. Trust relationships represented by certificates are transitive. Each node maintains a local certificate repository, and performs the public key authentication via a chain of certificates. Coming back to the above example, when a node knows/trusts *B*, that means that it has the public key of *B* and, therefore, it can decrypt the certificate that *B* issued for *S* that certifies the correct binding of *S'* public key and its identity. However, there is no guaranty that the certificates in the certificate chains are valid. A number of security threats arise when malicious nodes fabricate invalid certificates in the chain.

[16] propose a self-organized public key management where certificates are stored and exchanged by the nodes. The main problem of this scheme is its large overhead for storing the approximate global certificate graph. To solve this problem, [17] propose an on-demand public key management solution. In this scheme all certificates need to be issued and trusted locally. A certificate chain can be obtained hop-by-hop, as long as a route is discovered between source node and destination node. [18] propose a solution to cope with the malicious nodes. However, they assume the existence of a web of trust.

Without the assumption of a web of trust, a self-organized ad hoc network is an unsupervised and uncertain environment where nodes do not have any information about the trustworthiness, or untrustworthiness, of others. Therefore, via a trust management system nodes learn how to rely on the authenticity of other participants and how to avoid malicious nodes.

Each trust model is composed of two main components: a *trust computation* model and a *trust evidence distribution* system. The distribution part is the basis of the computation part. In this study we propose an identity trust model based on bio-inspired ant colony systems [19]. In ant colony systems the main interaction principle is stigmergy, which means that the trace left in the environment by an action, called pheromone, encourages the performance of the next action by the same or different agent (ant). The dynamic nature of ad hoc networks, triggered by the mobility and the changing behavior of nodes, makes ant colony optimization an appropriate choice for the autonomous authentication model.

1.1 Contribution of the Thesis

In this work we address following issues:

- We propose a trust model with a focus on identity trust as the scope of trust in self-organized mobile ad hoc networks. In most trust-based systems it is assumed that each entity has a unique identity. We propose an autonomous trust-based public key management system. In our proposed scheme each node creates its own public-private key pair, issues certificates to neighboring nodes and reactively performs the authentication process. Each node stores the trust level of its neighbors locally.

- Our bio-inspired scheme is based on an ant colony system. Reactively, a node performs public key authentication by sending out ants towards the target node. The task of ants is to find the most trustworthy certificate chain. A forward ant, who reaches the destination, will be transferred to backward ants and traverse the same path of the forward ant back to the source node. Through building a certificate chain, backward ants leave traces of pheromone along the path representing a path trust. Using the reinforcement method, trustworthy nodes become highlighted through the honest certificate chains. By integrating the reinforcement method of ant colony optimization with an incentive mechanism, we attain both trust computation and evidence distribution. Nodes belonging to an honest certificate chain will be rewarded; nodes along a certificate chain containing malicious nodes will be punished. In this regard, despite misbehaving nodes each node can make a suitable decision about the validity of the public key of a target node.

- Evaluation of a simulation of the proposed scheme attests its robustness against malicious behavior and topology changes caused by mobility.

1.2 Thesis Overview

This dissertation is organized as follows: First *soft security* and its corresponding mechanisms are explained in chapter 2. This chapter presents a common framework of trust in communication systems. Trust models in MANETs and their components are presented, followed by a discussion of identity trust management as the scope of trust in this thesis. Chapter 3 identifies and discusses different authentication mechanisms in mobile ad hoc networks. Furthermore, it explains our self-organized key management model for mobile ad hoc networks. Relevant security threats of autonomous key management model are also presented.

In chapter 4 bio-inspired learning mechanisms are briefly explained; among those, the ant colony system is reviewed in more detail. Existing ant colony optimization (ACO) approaches in mobile ad hoc networks are presented and the characteristics of the proposed ACO model are stated. Chapter 5 describes our proposed autonomous authentication design in more detail. Chapter 6 presents the experimental study of the model and shows the simulation results. Finally, chapter 7 provides the conclusions and presents future research work.

Look into yourself to find the reason
for everything that happens to you.
-Ostad Elahi

Chapter 2
Soft Security Mechanisms in Mobile Ad-hoc Networks

The goals of security in information- and communication systems are authentication, confidentiality, integrity, access control or authorization, non-repudiation, availability and privacy. Provision of these security services is a very challenging issue in mobile ad hoc environments due to the characteristics of these networks. In ad hoc networks there are no central trusted parties and every entity has simultaneously the role of both communication services, namely, provider and consumer. In the absence of infrastructure, nodes have to rely on their neighbors to provide data for local processing and to route their data. However, in such cooperative environment there is no guaranty that all participants in the network perform their tasks correctly. Malicious behaviors like eavesdropping, impersonation, modification, selfishness etc. can cause unexpected events and cripple the effective functionality of the network. There are security challenges that are not met by traditional security approaches; appropriate and efficient security solutions are necessary in order to encourage the nodes to cooperate correctly and to cope with misbehaving participants that disobey the communication protocol.

In general, security mechanisms are divided into two categories [1]:

- *Hard security* mechanisms
- *Soft security* mechanisms

Traditional security mechanisms aimed at protecting resources from malicious users by restricting access to the resources to only authorized users. These approaches are called *hard security* mechanisms. In contrast, *soft security* mechanisms emerged to provide security services based on social control. Prior to the description of soft security approaches in the subsequent sections, the following subsection presents security challenges in ad hoc environments and possible attacks that these challenges may lead to.

2.1 Security Challenges in Mobile Ad Hoc Networks

The nature of ad hoc networks makes them vulnerable to different kinds of attacks. In the following, ways by which security can be breached in these networks are presented:

Unreliable wireless channel: the wireless channel is unreliable and messages can be eavesdropped. Also, hidden node problems may occur.

Limited resources: the limitation of computational resources and battery power make these networks vulnerable to malicious behavior. There is a tradeoff between security and resource consumption. A higher level of security can only be achieved by appropriate costs of computational, power and memory resources.

Absence of infrastructure (decentralized): regarding the resource restriction, a single node cannot play the role of a central certification authority which controls the security of the whole network. Therefore, undertaking the security of the network by all participants, cooperation among nodes is unavoidable. Although this nature of ad hoc networks leads to a fault tolerant environment and prevents the single point of failure, there is no guarantee that all nodes act properly in cooperation.

High mobility: by topology changes, centralized authorities may become unreachable. Frequent topology changes also cause frequent link breakage and, consequently, sophisticated routing protocols with additional security challenges.

In [20] two different kinds of security attacks are considered in mobile ad hoc environment: *passive* and *active* attacks. In a passive attack the attacker executes the attack without actively initiating a malicious behavior. For instance, observing the channel and monitoring the activity and connection of other nodes, the passive attacker can recognize the crucial node in a special function like routing. Another example is *selfish* behavior in which a passive attacker denies forwarding incoming messages to prevent power loss. *Eavesdropping* is another example of a passive attack. An Active attack mainly occurs after a passive attack.

Performing an active attack, the attacker has to invest some of its energy. Basically, active attacks in ad hoc networks can be classified into three groups: integrity, masquerade and tampering (Fig. 2.1).

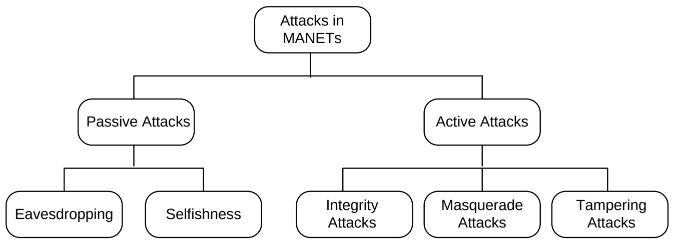


Fig. 2.1 Classification of attacks in MANETs [20]

Active attacks can range from deleting messages, injecting invalid messages to impersonating other nodes; consequently to violating integrity, availability, authentication and non-repudiation.

In *integrity* attacks, malicious nodes by *modification* aim against the integrity of message- or routing information. By this attack, a malicious node can, for example, modify the routing information and redirect the traffic to a different destination or increase the communication delay by computing longer route to the destination. By sending fake routing packets the attacker can set up *a Blackhole* attack. To launch this attack, a malicious node first analyzes the routing information by using eavesdropping traffic information. Subsequently, this node lies and announces that it has a route to destination in discovery phase of the routing protocol. Finally, all packets are transferred to the malicious node which then swallows them by discarding all packets.

Masquerading attack involves: *Spoofing*, *impersonation* and *Sybil* attack. In *Spoofing*, a malicious node modifies its identity (e.g., MAC or IP address) and can appear as an honest node; its purpose is a malicious action, e.g., advertising incorrect routing information, creating loops in routing computation to cause unreachable nodes or partitioning the network. In an *impersonation* attack, the malicious node pretends to be another node in the network in order to access resources under that node's name, or to be able to eavesdrop the messages sent to that node. Another dangerous attack is a Sybil attack [21]. The Sybil attacker pretends to have multiple identities to abuse the system.

In a *tampering* attack, often called *fabrication* attack, a malicious node generates forged routing messages. For example, an attacker rapidly and frequently spreads routing messages through the network in order to disable authorized routing messages. One possible example of such an attack is *Denial of Service* attack in which resources are unavailable for nodes in the network.

Due to the characteristics of the mobile ad hoc networks, traditional, centralized security mechanisms are not applicable. Soft security mechanisms are applied locally at each node in order to monitor the behavior of other participants, to detect malicious nodes and to isolate them from the network's functionality. In the following these security mechanisms are explained.

2.2 Soft Security vs. Hard Security Mechanisms

In *hard security* approaches, such as passwords, there is a special and centralized entity which is responsible for security provision and controlling, like a central authority used for authenticating users in a network. These approaches prevent unauthorized users from accessing the resources.

However, if a user finds a way to bypass the security provision, the whole data set will be revealed (Fig. 2.2). These approaches are also vulnerable to unreliable service providers. If a service provider acts deceitfully by providing false information, the users will not be able to protect themselves.

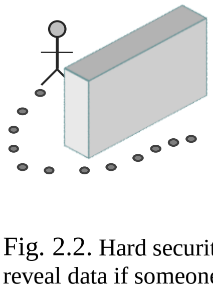


Fig. 2.2. Hard security will reveal data if someone finds the secret [1]

As a consequence, *soft security* mechanisms have been proposed to mitigate these drawbacks. In these approaches it is not only a single entity that is delegated security provision tasks; rather, all participants in the network are responsible for providing security (Fig. 2.3).

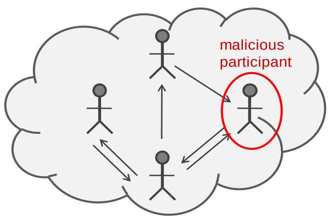


Fig. 2.3. Soft security allows entities to interact with one another as long as they behave nicely [1]

Like in a social network, malicious behaviors are not prevented; the existence of an unwanted intruder in the system is accepted, but the idea is to identify intruders and isolate them from service providing and consuming. There shall never be a key that uncritically opens up all locks on the system [1].
In social control-based *soft security* mechanisms, the notion of *trust* obtains significant importance for security provision. Trust, as an important aspect of human life, plays also a crucial role for communication security.

2.3 Framework of Trust

In [22] the author presents a unified taxonomy, a framework of trust. This framework includes topics like the notion or definition of trust relationship, properties of trust, type or classification of trust and a trust management model. In the following, an explanation of each topic is presented.

2.3.1 The Notion of Trust

Trust is expressed in three diversity dimensions [23]. It is normally presented as a relationship between two entities: *trustor*, the entity that trusts a target subject or entity, and *trustee* which is the trusted entity. The purpose of trust is another main component of a trust relationship, called the *scope* or the *domain* of the trust relationship. It means that a trust relationship applies to a specific purpose or domain of action, such as "being authentic" in the case of an entity's trust in a trustee's cryptographic key [24].

There is a variety of definitions of trust in the literature [25]. In the Oxford English Dictionary (OED) trust is defined as:

Definition 2.1- trust is confidence and strong belief in the goodness, strength, reliability and honesty of something or somebody.

Trust, in form of 'cooperation', is defined in various ways in sociology and psychology literature.

Sociology-based trust is a form of reliability of somebody in a cooperative environment, as presented by Gambetta [26]. He defines trust as follows:

- *When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him.*

The definition of trust is formulated in [2] as follows:

*Definition 2.2 - Reliability Trust***:** Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which A's welfare depends.

Morton Deutsch defines trust, in psychological terms, as follows:

- *If an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial (Va+) or to an event perceived to be harmful (Va-);*
- *He perceives that the occurrence of Va+ or Va- is contingent on the behavior of another person; and*
- *He perceives that strength of Va- to be greater than strength of Va+.*

If he chooses to take an ambiguous path with such properties, I shall say he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice.

This definition implies that trust depends on an individual and his/her benefits. Choosing an ambiguous path is an uncertain situation which could be harmful, or harmless, for the trustor. In this definition trust is linked to the perceived harmfulness and beneficial cost of the path [27].

Based on this notion of trust, author in [2] presents another definition of trust.

Definition 2.3 - Decision Trust: trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

Another important concept used in soft security mechanisms, closely linked to the notion of trust, is *reputation.* The definition of reputation in the OED is:

Definition 2.4 – Reputation is what is generally said or believed about a person's or thing's character or standing.

Reputation can be considered as measures of trustworthiness based on the ratings received from other members of a network.

In contrast to trust, which is a personal opinion, reputation is a global quantity derived from the entirety of members and is visible to all members in a network. Although trust and reputation are the foundation of the soft security approaches, there is a clear and important difference between them. The following two statements illustrate the difference between trust and reputation [2]:

- *"I trust you because of your good reputation".*
- *"I trust you despite your bad reputation".*

The first statement means that the relying party is aware of the trustee's reputation, and bases his trust on it. The second statement reflects that the relying entity has some direct private knowledge about the trustee, and that this information overrules the reputation of the trustee. This comparison shows that trust is a subjective opinion based on several factors that carry different weights. Direct and personal experiences carry more weight than reputation and referrals received from others; however, in the absence of personal experience, trust has to be built upon trustee's reputation.

Applying the notion of trust in a formal model in a meaningful way leads to different computational concepts.

2.3.2 Properties of Trust Relationships

Trust relationship properties, cited in (Jøsang et al., 2007), are as follows:
Subjective: a trust relationship is a subjective belief of someone in the character, ability, strength, reliability, honesty or truth of someone or something. This subjective nature of trust demonstrates that every trustor personally builds up an individual opinion about the trustworthiness of the trustee.
Specificity: trust is specified for a particular context. For example, service providing.
Asymmetric: trust relationships, in general, are not symmetric**.** If *A* trusts *B*, it does not imply that *B* trusts *A*. Therefore, trust relationships are modeled as unidirectional relations.
Direct and indirect trust: trust relationships can be considered as direct trust or indirect trust. Direct trust, or *functional trust*, demonstrates the opinion of the trustor that the trustee has the specified capability. Indirect trust, or (referral/recommender trust), on the other hand, represents the opinion of the trustor that the trustee will supply honest and useful recommendation for this capability.

Transitivity: transitivity plays a major role in trust models. This property leads to the derived trust relationship based on direct trust and recommendations. In , if Alice has direct trust in Bob in a particular context, and Bob has direct trust in Claire, then Alice can trust Clair indirectly via the referral received from Bob.

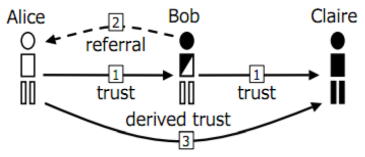


Fig. 2.4 Trust transitivity principle [2]

Time variability: trust may change over time, either due to new experiences or due to an activity of the trustee.

Like in social life, trust relationships for security provision need to be built over time. In order to make a subjective judgment, a trustor needs to learn from the trustee’s history and past experiences. The subjective opinion of a trust relationship can be derived, over time, from the combination of referrals and individual experiences. However, in order to prevent dependence and loops, only referrals based on first hand experiences, and no other referrals, will be taken into consideration.

2.3.3 Trust Classes

Jøsang [2] presents a classification of trust which describes different type of trust.
***Provision trust*:** provision trust describes the relying party's trust in a service or resource provider. It is relevant when the relying party is a user seeking protection from malicious or unreliable service providers.
Access trust: Access trust describes trust for the purpose of accessing resources owned by, or under the responsibility of, the relying party.

Delegation trust describes trust in an agent (the delegate) that acts and makes decisions on behalf of the relying party.

***Context trust (System trust)*:** Context trust describes the extent to which the relying party believes that the necessary systems and institutions are in place in order to support the transaction and provide a safety net in case something should go wrong. Factors for this type of trust can, for example, be critical infrastructures, insurance, legal system, law enforcement and stability of society in general.

***Authentication trust (Identity trust)*:** authentication trust describes the belief that an agent's identity is as claimed. Trust systems that derive identity trust are typically authentication schemes such as PGP [28].

The classification of trust defines the third dimension of trust. It means that the class of the trust, which expresses the purpose of the trust, defines the specific *scope* of a trust relationship.

As the authentication trust is the fundamental type of trust on which other types are built, it belongs in a different layer. Authentication trust belongs in the first layer and all other trust classes belong in the layer above it (Fig. 2.5). In general, other types of trust cannot exist in the absence of authentication trust.

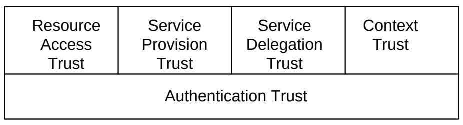


Fig. 2.5. Trust classes in two main layers

Authentication trust is the condition for trusting an entity behind an identity; however, it does not mean that the real entity's identity must be known. An authorized, anonymous entity can also be trusted for accessing services.

2.3.4 Trust Management

Trust management characterizes a self-organized system used to form, evaluate, maintain and exchange trust information about the entities in a network. The definition of trust management presented in [24] is as follows:

Definition 2.5 - Trust Management is the activity of creating systems and methods that allow relying parties to make assessments and decisions regarding the dependability of potential transactions involving risk, and that also allow players and system owners to increase and correctly represent the reliability of themselves and their systems.

In [29] a trust management model is proposed that includes the different aspects of human trust (Fig. 2.6). The main components of the model are:

- Trust formation
- Trust dissemination
- Trust evolution

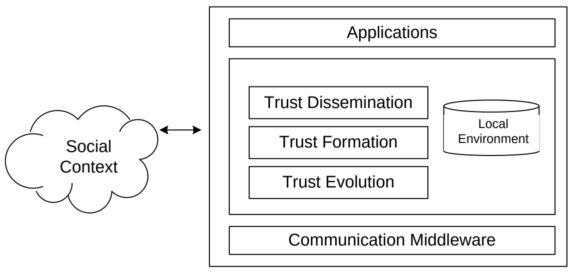


Fig. 2.6. Overview of trust management model [29]

As the picture shows, these components are located between applications and a communication middleware that enables the agent to interact with other agents in the system. In this model, the data needed for trust calculation is both direct experiences (e.g., entities' history) and recommendations. In general, the process in the *trust formation* phase enables a trustor to establish trust in a trustee. It refers to the process of determining whether a trust relationship can be created or not. All gathered data, processes, like decision making, and final opinions are kept locally on each individual entity. Various computation models are applied for the trust formation phase. The presented computational approaches in [2] involve: simple summation or average of ratings; weighted average of ratings; Bayesian systems; discrete trust models; belief theory; fuzzy models; flow models.

Recommendations coming from other entities will be propagated by means of the *trust dissemination* component. This component plays an important role in providing reliable recommendations for the trust formation part. If a trustor, *S*, wants to form a trust opinion about a trustee, *D*, but has not kept locally enough trust information about the target's entity, the trust dissemination function should be activated and go through the following steps:

- *Step 1*: In a first step *S* makes a request to gather evidence about *D*'s trustworthiness. Evidence can be credentials that indicate D's trust value in the point of view of other entities that have information about the target node *D*.
- *Step 2*: In order to reject malicious recommenders who reported false credentials about the target node, *S* requests to receive further recommendations about *D*. In this step more information may be collected.
- *Step 3*: Receiving different recommendations, *S* computes its subjective trust opinion about the target node using the trust formation module.
- *Step 4*: In this step, upon the calculated trust level, interaction between *S* and *D* may or may not take place.

In general, the more recommendations collected in this phase, the faster and easier the discovery of malicious nodes that spread fake recommendations.

Trust evolution is a continuous self-adaptation of trust information in the entity's local area. In this phase trust values are updated during the life time of a system. Trust evolution maintains the trustworthiness of entities as a service provider and as a service recommender. Updating the trust values of entities after performing an interaction is based on past experiences; trust updating for recommenders is based on the *conflict detection* mechanism. A conflict detector detects malicious nodes and prevents them from attending in cooperation.

Due to the characteristics of ad hoc networks, trust management and decision methods to determine the trustworthiness of an entity should be localized and fully distributed, since considering a trusted third party to locate, calculate and disseminate trust values is not feasible in such an environment. Regarding dynamic character and quick topology changes, trust management in mobile ad hoc networks should support fast, online and short trust evidence modeling among the participants and should be independent of pre-established infrastructures.

2.4 Trust and Network Security

The application of *soft security* mechanisms has emerged mainly for online service provision where the service consumer often does not have enough information about the service provider. These mechanisms are also applicable in different network security domains. Network security and trust have a direct relationship. When communication of two parties in a system is more secure, the level of trust between entities will increase; on the other hand, having a stronger trust relationship between participants of a network results in an increase of the network security. In [30] security services include:

- *Authentication*: ensures that the other end of a connection, or the originator of a packet, is the node that is claimed.
- *Access control*: prevents unauthorized access to a resource.
- *Confidentiality*: protects overall content or a field in a message. Confidentiality can also be required to prevent an adversary from undertaking traffic analysis.
- *Integrity*: ensures that a packet is not modified during transmission.
- *Availability*: mainly targets DoS attacks and is the ability to sustain the networking functionalities without any interruption due to security threats.
- *Non-repudiation*: proves the source of a packet. In authentication the source proves its identity. Non-repudiation prevents the source from denying that it sent a packet.

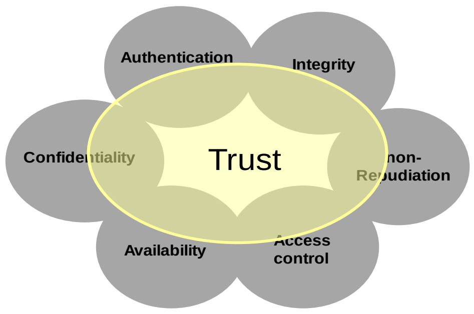


Fig.2.7. Notion of trust is applicable in different network security domains

Fig.2.7 represents the trust and network security relationship.

2.5 Trust Models in Mobile Ad Hoc Networks

Trust and reputation system in a wireless ad hoc network can be regarded on two levels: *individual level trust model* and *system level trust model* [31]. In the *individual level trust model*, these systems should enable the nodes to predict the behavior of other nodes. In self-organized, wireless ad hoc communication nodes apply trust and reputation mechanisms in order to select their trustworthy neighbors for interacting with them, e.g., routing a message, getting required information, etc.

In the *system level trust model* the security system includes a framework that, in addition to the trust and reputation evaluation model, also contains a protocol for nodes to interact. The protocol consists of mechanisms for rewarding good behaviors and punishing bad behaviors. With this protocol, based on the behavior of the nodes, trustworthiness can be earned or lost. The system’s rules of encounter, which induce the nodes to behave in a trustworthy manner, are known as system level trust.

Fig. 2.8 shows the architecture of the trust and reputation models in wireless communication systems.

2.5.1 Individual-Level Trust Model

The absence of a central trusted authority causes the nodes in an ad hoc environment to control and modify their interaction strategy with other nodes; they can thus maximize their service gain and the ability of distinguishing the reliability of direct information received from other nodes. Prior to interacting with other nodes, a node should be able to estimate the probability of a successful interaction. To achieve this goal, one option is to interact with the whole community and to gather first- hand information about the respective nodes and, based upon the experiences, the trustworthiness of each node would be calculated. This mechanism needs an intensive interaction and a long estimation period; the assessment process can be sped up when a node utilizes the information gathered by other nodes (Fig. 2.8, left side). Therefore, as there is no central authority to vouch for the trustworthiness of the nodes, nodes share their first-hand information, gathered from direct interactions with others. In the evidence space a node combines first-hand and second-hand information. Then aggregated information by using a predefined mapping functions will be transformed to the trustworthiness level of a subject node. Based on the calculated trust level, a decision is made whether to interact with the subject node. After the transaction, based on the assessment of satisfaction or dissatisfaction of the subject node's performance during the interaction, the first-hand evidence is updated and forms the future trust opinions.

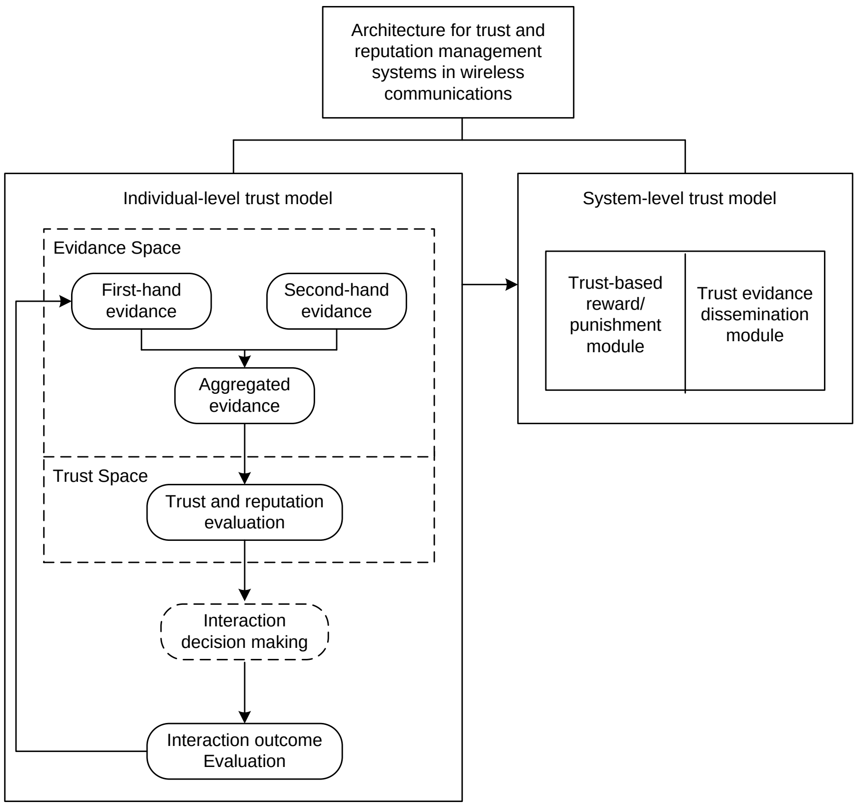

Architecture for trust and reputation management systems in wireless communications
Individual-level trust model
Evidance Space
First-hand evidance
Second-hand evidance
Aggregated evidance
Trust Space
Trust and reputation evaluation
Interaction decision making
Interaction outcome Evaluation
System-level trust model
Trust-based reward/ punishment module
Trust evidance dissemination module

Fig. 2.8. Architecture of the trust and reputation systems in wireless communication systems; consists of two individual- and system-level trust and their components [31]

Although second–hand information speeds up the estimations of nodes' trustworthiness, there is no guarantee that the information about a node, disseminated by other nodes, would be accurate. Aggregated false second-hand evidence can affect the trustworthiness of a subject node and, consequently, the final interaction decision. Two kinds of attacks threat the credibility of the evidence gathered from other nodes: *badmouthing,* where false evidence causes a reduction in the trustworthiness of a node; *ballot stuffing,* where the false reported evidence increases the evaluated trustworthiness of a node.

RFSN [32] proposes an individual-level trust model. Wireless sensor networks, as a kind of ad hoc networks, are deployed to automatic data collection of phenomena of interest. However, the accuracy and reliability of the gathered data are questioned. Data integrity is vulnerable to failure of nodes and the final fused data is affected by corrupted sensor measurements. RFSN is a framework for wireless sensor network in which nodes maintain others' reputation via a watchdog mechanism. The responsibility of watchdog module is monitoring the action of others and characterizing them as cooperative or non-cooperative. Based on the calculated reputation over time, a node – at the collaboration time – cooperates only with cooperative nodes which it trusts. For trust and reputation representation, updates integration and evaluation it employs a Bayesian formulation. In RFSN *badmouthing* attacks are prevented by propagating only positive second-hand information. It combats *ballot stuffing* attack by considering the reputation of the witness node to weigh its reported evidence.

2.5.2 System-Level Trust Model

Individual-level trust models usually deal with the computational methods to evaluate the behavior of a node based on its history. This model meets the requirement to decide whether a node is malicious or not. However, this individual-level will not prevent a malicious node from continuing its misbehavior. A system-level trust model should include punitive and incentive mechanisms to prevent misbehaving nodes by punishing them and by rewarding the nodes that act properly according to the rules of protocol. Figure 2.8, right side, illustrates a system-level trust model that contains reward and punishment modules to isolate misbehaving nodes from the normal functionality of the network; a trust evidence dissemination mechanism to ensure sharing of a node's observation throughout the network is also included. In [33] it is proved that in non-cooperative network the absence of any punishment mechanism converges to an equilibrium state. A variety of mechanisms has been proposed to induce cooperation in the network. Considering different contexts of trust, these mechanisms are applied in different domains. Misbehaving that affects routing and packet forwarding may range from *selfishness* to *malicious* behavior. In order to save energy and have a longer life time, it might be the case that nodes in an ad hoc environment behave *selfishly* when they are requested to rely packets of other nodes;

their aim is not to directly damage other nodes. On the other hand, *malicious* nodes, saving their own power life, intend to damage the routing network operations.

For example, in [33] a mechanism for packet forwarding, based on game theory, is proposed. Trust- and reputation-based incentive/punitive mechanisms are proposed to facilitate the routing process and to prevent selfish behavior.

COFIDANT [34] protocol is a trust- and reputation-based system that copes with misbehaving in packet forwarding. It feeds the system-level trust with the nodes' reputation, calculated in individual-level trust, to punish nodes with low reputation values. It is assumed that each node is equipped with a monitoring component to detect misbehaviors of neighbor nodes. Suspicious activities are reported to the reputation system of the observing node to evaluate their significance and, consequently, to increase the trust value. If the trust value exceeds the threshold, an ALARM message will be sent to either the source of the route or to the friends. When receiving such an ALARM message, it will be passed to the trust management module to evaluate the source of the message. If there is sufficient evidence that the reported node is malicious, the information will be sent to the reputation system for further evaluation. Sufficient evidence is provided when the ALARM message is generated either by fully trusted or several, partially trusted nodes that have reported the same about the subject node. One drawback of the system is the permanent exclusion of misbehaving nodes. If there is a mistake in the watchdog- or reputation system, or a badmouthing attack has been successful, a well-behaving node may be permanently excluded from the network.

CORE [7] is a reputation-based routing protocol dealing with selfish nodes. It also employs a watchdog mechanism to monitor the behavior of the other nodes in the network. However, CORE equips the nodes with different groups of watchdogs; each of them, specified by the system designer, is planned to evaluate certain functionality in the network. When a node needs to monitor the correctness of the execution of a function by its neighboring node, it triggers the watchdog related to that action.

CORE classifies all entities in the network as service *provider* or *requester*. The service can be any activity in the network that needs cooperation among entities. When a requester asks for a service, the provider first checks the requester's global reputation. If it is less than the threshold value, the provider will not execute the requested function and possibly reduce the reputation of the requester. At the same time, the requester applies its watchdog while requesting the service and evaluates its behavior.

In order to disseminate reputation of a node through the network, CORE assumes no collusion among nodes and therefore avoids ballot-stuffing attacks. It prevents badmouthing attacks by allowing only positive reputation information to be disseminated. A single deviation from the norm does not have great impact on the node. Only multiple observations about the misbehaving of a node provide sufficient evidence to classify the node as a misbehaving node. Once a node is labeled as misbehaving, its request for getting services from others is denied. It can only take the role of the service provider but not of the service requester. Compared to CONFIDANT, punishment is temporary and a misbehaving node can still increase its reputation by consistently providing correct services to other nodes. When its reputation value exceeds the threshold value again, its service requests will not be denied.

SORI [35] is a reputation-based scheme to identify and penalize selfish nodes in a network. The model assumes a promiscuous communication mode; with it a node is capable of overhearing the transmission of its neighbors. The trustworthiness of node X evaluated by node N is as following:

$$RF_N(x)/HF_N(X) \tag{2.1}$$

where $RF_N(x)$ is the total number of packets that node N has transmitted to X for forwarding and $HF_N(X)$ is the total number of packets that have been forwarded by X and noticed by N. Punishing selfish nodes, based on one's own observation, is not effective. As a result, the model lets the neighbors share their observations with their neighbors. Each node periodically updates its *local evaluation records* ($LER_N(X)$) for its neighbors. If there is a significant change in the trustworthiness of a node, the observation will be broadcasted to all neighbors. If a node receives reputation broadcasts from different nodes, it aggregates all received information. In aggregation, observations are weighed by the credibility of the broadcasting nodes. The credibility values of broadcasting nodes are considered the same as their trustworthiness in the context of packet forwarding. SORI assumes that nodes do not lie when sharing their observations.

A punishment action is defined probabilistically by dropping the packets that originated from the selfish node. The probability of dropping depends on the reputation of the node. In this model selfish nodes are not completely isolated from the communication. A node still has a chance of increasing its reputation value by consistently cooperating with other nodes. Once its reputation value exceeds the threshold value, it will not be punished further. Since in SORI nodes exchange their information with their neighbors only if the trust value falls below a threshold, the overhead is less than the overhead of the CONFIDANT protocol [31].

ABED [36] is a scheme for distributing trust certificates. It is an ant-based evidence distribution algorithm with the principle of stigmergy for communications. Entities in the system communicate with each other by modifying the environment and without direct interactions. The model consists of two parts: trust computation and trust evidence distribution. The first component, which is not addressed in the model, evaluates the trust level of each entity in the network, based on the history of behavioral data.

ABED assumes a central trusted party that issues and signs trust certificates for all participants prior to the setup of the network. Certificates can contain different information, depending on the trust model. The second part is system-level trust which is responsible for trust evidence distribution; this part has an important role, because it provides the input data for the trust evaluation part. When a certificate is required, several ants are sent out to find the target node. Once a forward ant has found the required destination, a backward ant is generated. It traverses the exact path of the forward ant and reinforces the traversed path towards the source node by tracing pheromone on the path. Each node on the path of the backward ant stores the certificate, i.e., trust certificates get distributed and the trust values are updated each time the backward ants visits the nodes. The density of pheromone decides whether the certificates are valid or not.

2.6 Identity Trust Management

Trust, in general, is interpreted as the relation among participants in various contexts. The aim of the presented protocols is to detect authorized entities by establishing a trust relationship. For example, by developing a trust relationship with other entities, authorization is defined as a decision policy providing them services or assigning them access rights to perform an action.

In contrast to these works, which focus on *service provision trust*, in this thesis we propose a trust model in the context of *identity trust*. *Identity trust* or *authentication trust*, as a fundamental security service, is a trust relationship that is established toward an entity when the identity that it claims really belongs to this entity. As a result, other security services, like access control, confidentiality and non-repudiation will be achieved. Authentication is the verification of an entity’s identity which can be achieved with a password, a trusted central authority, or by means of a certificate.

Due to the characteristics of ad hoc networks, with the aim of anywhere and simple establishing communication between mobile devices, having trusted central authorities is not applicable. Therefore, self-organized authentication mechanisms are a better fit to mobile ad hoc networks. We consider a distributed authentication mechanism in which each node is responsible for authentication mechanism. All participants create their identifier by their own, and via an authentication process it is assured that the communicating entity is the one that it claims to be. It is not important which identifier a node creates for itself; important is acting faithfully in regard to the created identifier; the created identifier, should matches to what it claims to be and by these means introduces itself to the other participants in the network.

Regarding the lack of a trusted party to perform the authentication process, an *identity trust management* system is needed. The purpose of an identity trust management is to make each created identifier available to others and protect these identifiers from tampering. The components of an identity trust management are trust creation, trust dissemination and trust evolution. The condition of creating an identity trust relationship, e.g., from node *A* towards node *B*, is the guarantee that *B* assures the matches between its identifier and what claims to be its identifier. The circumstances of identity trust dissemination can be explained in the following example. Node *A* has direct identity trust to node *B* and wants to know the identifier of node *C*, with whom *A* does not have direct identity trust. Node *B* has identity trust to node *C;* therefore, by asking node *B* node *A* can access the correct identifier of *C* if *B* provides valid witness about C’s identifier.

In order to protect identifiers when distributing and forwarding them across unsecure media, certificates are issued. A certificate represents a trust relationship between certificate issuer and the certificate subject. To check the validity and authenticity of a destination node’s identifier, chains of certificates are required. Chained identity certificates, for derivation of identity trust, are based on trust transitivity. The existence of a certificate and the application of trust transitivity properties are necessary for an authentication process, but are still not sufficient for a chain’s validity. There are some security threats when a certificate chain is invalid or created by attackers. For example, when a node’s identifier is changed but the certificate has not yet been updated; or in the case some certificate or identifiers have been compromised.

Through system-level trust, evidence is gathered as a form of different certificate chains from source node to a target destination. In the system level trust, this evidence is checked for validity. If the source node receives chains consisting of fake or mismatched certificates, it sends them punishment messages to penalize all nodes in the chain. All nodes of a chain, which have been proved to disseminate a correct certificate, will be rewarded. In our identity trust model we assume that nodes do not lie for punishing and rewarding message dissemination and do not send fake punishing/rewarding messages.

The most difficult thing in life
is to know yourself.
-Thales

Chapter 3
Authentication Mechanisms in Mobile Ad-hoc Networks

Two major components are required in order to provide security: data transformation techniques (encryption and decryption); and keying information.
The general model of security in mobile ad hoc networks has four basic tasks for security mechanisms [37]:

1. The design of the security algorithm.
2. Generation of key materials (secret keying material or private public key pair keying materials).
3. Distribution of keying material.
4. Protocol which will achieve the required security services; for the participants to follow.

The first and the second tasks deal with the mathematical aspect of the cryptographic algorithms applied for security service provision. This part uses widely accepted, existing cryptographic techniques to ensure confidentiality, authentication, integrity of the routing message and non-repudiation of the origin of the message. Figure 3.1 shows a general security model in mobile ad hoc networks. A message, *M*, is to be transmitted from a source S to a destination node D through intermediate nodes in the network. Security provision is needed in order to transfer data from source to destination safely via multi-hop route, involving secondary entities.

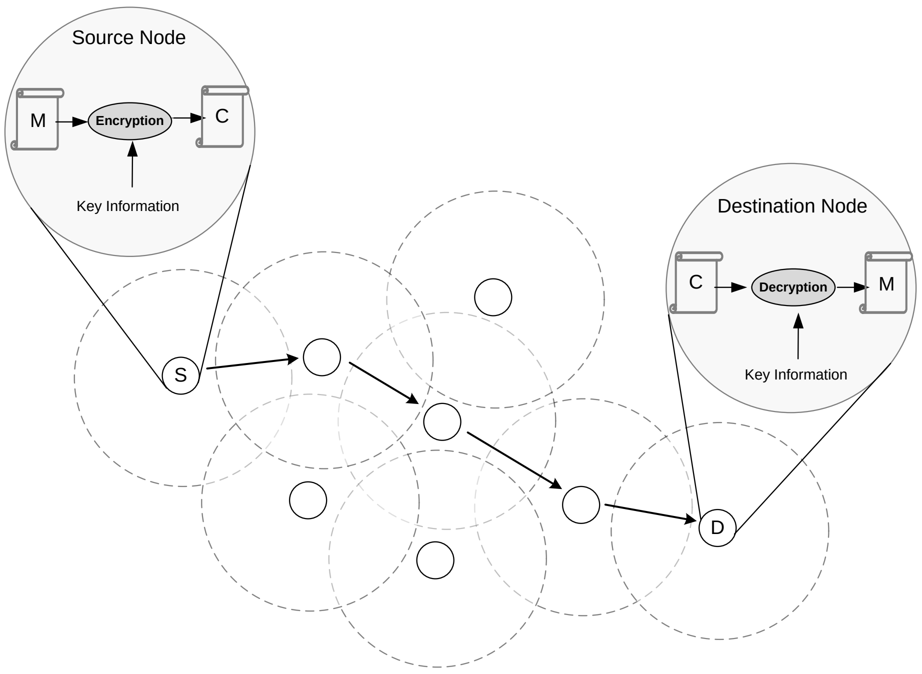


Fig. 3.1. Security model in mobile ad hoc networks

The focus of this thesis is upon the third and the fourth tasks: the establishment of security protocols in the mobile ad hoc networks. The work is concerned with the distribution of the keying materials issued in a distributed and autonomous manner. It also proposes a mechanism to make the participants obey the protocol despite a central security controller.

3.1 Authentication as the Heart of Security Services

Authentication is significantly important for providing secure communication in any network. Authentication can be achieved if a key agreement or key distribution scheme is available and works properly. Authentication is a process that contains an *authenticator* who aims at authenticating and, consequently, communicating with a *supplicant* using an *authentication protocol* to verify *credentials* presented by the supplicant as proof of its identity. Authenticator is an entity that makes authentication decisions in order to provide access control. Supplicant is an entity that can access some resources after being authenticated via an authenticator. An authentication protocol is a sequence of massages exchanged between authenticator and supplicant. A credential is an identifier that can be used to authenticate an entity with high certainty [38]. In [12] different definition of authentication is presented.
Definition 3.1: Data origin authentication is a type of authentication whereby a party is corroborated as the (original) source of specific data created at some (typically unspecified) time in the past. By definition, data origin authentication includes data integrity.

Definition 3.2: Message authentication is a term used analogously with data origin authentication. It provides data origin authentication with respect to the original message source and data integrity.

Definition 3.3: Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired).

Definition 3.4: Key authentication is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key. Key authentication differs from actual possession of such a key by the claimed party. Therefore there is another concept which is key confirmation.

Definition 3.5: Key confirmation is the property whereby one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key.

3.2 Identification and Entity Authentication

Identification, or entity authentication, and less frequently identity verification is a technique that allows one party, called *verifier,* to gain assurance that the identity of another party, called *claimant,* is confirmed. The aim of identification techniques is to prevent impersonation; the outcome of an identification protocol, from the point of view of the verifier, is acceptance or rejection of the claimant's identity. More specifically, the objective of an identification protocol is as following: in case of having two parties, A and B, A is able to prove itself as an authentic party to B; then B will complete the authentication process by verifying the correctness of a message which shows that A is in possession of corresponding keying materials.

Entity authentication and entity identification can often be used interchangeably. Message authentication and entity authentication are closely related; a major difference between them is on the time factor. Providing message authentication does not guarantee that message authentication was created immediately prior to sending the message. Entity authentication, in contrast, involves two parties communicating actively. Therefore, to provide this requirement some kind of clock or timeliness is needed. An identification protocol is a *real-time* process, i.e., it ensures the correctness of the authenticity of the party at the time of protocol execution by carrying out some action during the protocol execution.

3.3 Basis of Identification

Traditionally, entity authentication techniques are based on three factors [12]:

- *Something you know*. Examples like passwords, Personal Identification Numbers (PINs) and a secret/private key.

- *Something you have*. This is a physical accessory, possessed by the owner, which acts as a passport. Examples include hardware tokens or chip cards which provide time-variant passwords.
- *Something you are*. This category involves methods that use physical characteristic (biometrics), e.g., fingerprints, palm prints, handwritten signature or voice.

But recently a fourth factor for authentication has been proposed in [15]:

- *Somebody you know*. This factor deals with the social network of the user. An example of this category comes from old-age practice: human authentication through mutual acquaintance. The concept has been transferred to the area of computer security: peer-lever certification and reputation networks.

This fourth factor leads to the notion of *vouching*, which is intermediate peer-level authentication for access control of a supplicant. This factor was inspired by human relationships for authentication (which is by no means new). The most common way of identifying (authenticating) acquaintances in social interaction is by introducing one person to another. In the realm of entity authentication, vouching is considered a process in which one intermediate party assists a second entity to be authenticated at the point of view of an authenticator party.
Many authentication mechanisms and key management systems for ad hoc networks are based on the vouching concept to fulfill the authentication process.

3.4 Key Management Techniques

In general, key management systems are responsible for distributing the key to a destination node in a secure manner.
Definition 3.6: key management is the set of techniques and procedures supporting the establishment and maintenance of keying relationships between authorized parties [12]. Key management is a basic requirement and plays a fundamental role for providing confidentiality, entity authentication, data origin authentication, data integrity and digital signature. The key management system supports:

1- Initialization of system users within the network
2- Generation, distribution and installation of keying material
3- Controlling the use of keying material within the network
4- Update, revocation and destruction and maintain keying material
5- Storage, backup/recovery and archive of keying material

In a communication environment for keying relationship in real-time, two parties (sender and receiver) are needed. The aim of key management is maintaining keying material at the face of threats, such as compromise of confidentiality of secrete key, compromise of authenticity of secret or public key, unauthorized use of secret or public key.

The goal of key management in *secret key* distribution (symmetric cryptography) is to guarantee that the secret key is shared among the authorized communicating entities securely. Secret keys need confidentiality, authenticity and integrity. There are some approaches for sharing a secret key like: key transport, key arbitration, key pre-distribution and key agreement [37].

Key management for public keys (asymmetric cryptography) does not need confidentiality, but authentication and integrity are required. Key management systems must prevent unauthorized use of keys, e.g., use of expired or invalid keys.

3.4.1 Key Management in Mobile Ad Hoc Networks

Traditional security provision mechanisms, used in conventional wired network, are not applicable for mobile ad-hoc networks. In MENETs, there is no trusted third party (TTP) or central authority (CA); the users can rely on it completely. However, networking and security provision are the responsibility of all participants. Not only special entities have this responsibility, but it is distributed across all nodes in the network. In the network initialization phase, entities have no prior relationship with each other. Therefore, a trust relationship will be established only after a network has been formed. The sporadic networking links are very prone to security attacks. Trust establishment is responsible for creating a secure network layer and honest links.

Frequent link breakage is the result of the mobile nature of such networks. Therefore, creating a trust relationship is challenging. The dynamic topology and the distributive nature of mobile ad-hoc networks yield a design of both routing protocol and trust establishment together, in order to handle reliable wireless communication.

Security research for ad hoc networks initially focused on secure routing protocols; however, most of the routing schemes neglect the crucial task of secure key management and assume there pre-existence and pre-sharing of secret and/or private/public key pairs [11].

This chapter focuses on presenting the existing key management schemes for mobile ad hoc networks. For the investigation of the key management methods following factors are important [39] [37]:

Availability: Availability plays an essential role in networks with changing topology. Authority members, who are responsible for keying services, should be easily accessible by other nodes in the network.

Security: Key management systems should ensure that no unauthorized node receives key material which can be used later as a legitimate member of the network. The system should have intrusion tolerance and should not succumb to a single or a few compromised nodes.

Scalability: Key management systems can effectively provide secure keying services in networks with dynamic changes in their size when nodes join and leave the network.

Robustness: Authentication- and key management systems should survive and be able to be completed despite faulty and malicious nodes that deliberately deviate from the protocol.
Efficiency: Key management solutions have high priority if they can provide successful and efficient secure keying services like key authenticity, key confidentiality, key integrity and key updates.

3.5 Authentication Mechanisms in Mobile Ad Hoc Networks

Providing security for an ad hoc network, authentication should be supported as the most important service, since other security services like confidentiality, integrity, non-repudiation and access control can only be provided when authentication has been performed properly. Authentication mechanisms can be classified in several ways. A classification of authentication protocols for ad hoc networks is presented in [38]. The authors categorize the existing authentication protocols in three groups that are based on: the type of credentials, the role played by nodes in the network with respect to authentication process and the phase when credentials are established.

3.5.1 Classification Based on Type of Credentials

Authentication protocols can be classified according to the type of *credentials* used for authentication. These credentials can be categorized into two main classes: identity-based credentials and context-based credentials.

3.5.1.1 Establishing Credentials

This classification is based on the time during which credentials are established. In some protocols credentials are established and distributed before node deployment, such as pair-wise keys that are pre-distributed to all nodes for future use. An example of such protocols is symmetric key based protocols in sensor networks. In a second category, credentials are established post-deployment, i.e., when a protocol relies on contextual information. The third category assumes that credentials are pre-distributed offline in the initialization phase; however, actual credentials used for authentication are derived from the initial credential post-deployment.

3.5.1.2 Identity-Based Credentials

In this class it is assumed that a *supplicant* possesses a unique identifier in order to prove its authenticity with high certainty. If it is assured that the supplicant possesses the key, the authenticator can be certain about the supplicant’s identity. Identity based credentials can be further classified according to the applied cryptographic[1] techniques

[1] - Cryptography is an ancient art back to circa 1900 BC. It is the science of converting data into secret code (encryption) and then translating code back into data (decryption).

and categorized into encryption based and non-encryption based. The classification is based on the number of keys that are employed for the authentication process:

- Encryption based
  - Symmetric cryptographic techniques/ secret key cryptography: use of single key for both encryption and decryption.
  - Asymmetric cryptographic techniques/ public key cryptography: use one key for encryption and another for decryption
- Non-encryption based
  - Hash function: Use a mathematical transformation to irreversibly encrypt information

Encryption identity-based credentials are used in order to verify a supplicant's possession of the key and so prove its identity. They are a piece of information produced and cryptographically signed by the supplicant's key. In order to verify the supplicant's identity, the authenticator needs either the same key (symmetric key cryptography) or the public key of the related private key owned by the supplicant (asymmetric key cryptography).

Credentials in non-encryption identity based are information that hashed by applying a one-way key based hash function and the key belongs to the supplicant. The authenticator verifies the claimed identity of the supplicant by regenerating the hash value. For this reason the authenticator should possess the same key and the hashed information.

- **Symmetric (secret key) cryptographic techniques:** In symmetric cryptographic techniques, or secret key cryptography, in order to provide authentication service a single shared secret key is employed among the nodes (pair of nodes that want to communicate with each other or all nodes requesting access to the network). Any key that possesses the secret key can authenticate itself by presenting the key and use network resources. Although in this kind of authentication communicational and computational overheads are low, a big difficulty is key distribution. Symmetric key based techniques are only appropriate for small scale networks. Furthermore, in case a secret key is shared between all participating nodes, if a single node is compromised, the whole network is compromised. In order to prove its identity and possession of a secret key, a supplicant signs a credential using the key. The authenticator is able to verify the supplicant's identity if it owns the same key.

- **Asymmetric (public key) cryptographic techniques:** Asymmetric cryptographic technique, also known as public key cryptography, uses a pair of keys (a public key, PK, and a private key, Pr) for each node in authentication processes. The private key is known only to the node to which it belongs or was issued to, while the public key of a node can be known to all participants. The two keys are mathematically related, although knowing one (public key) does not reveal the other key (secret key). In general, one key is used for encryption and the other for decryption. It is of no importance which key is applied first, but both keys are necessary for performing the work properly. In order to verify a supplicant's identity, the supplicant signs the credentials with its private key and the authenticator decrypts it with the supplicant's public key. Encrypted information with the public key of an authenticator can only be decrypted with its secret key. There are different reasons which make asymmetric cryptography suitable for mobile ad hoc networks. This cryptography mechanism is useful for providing authentication purposes together with information confidentiality and integrity.

Furthermore, for efficiency reasons and to reduce power consumption, asymmetric cryptography applies to exchange one time/temporary, symmetric key for secure communication.

3.5.1.3 Context-Based Credentials

In this class a unique contextual attribute is used in order to identify the supplicant with high certainty. In general, these credentials can be behavioral or physical. In behavioral-based contextual credentials, identifying and authenticating the supplicant are based on its pattern of behavior. The authentication process is done by monitoring the behavioral pattern of the supplicant with respect to certain functionality and by classifying it according to its performance. In physical-characteristic based, credential is the unique physical characteristic of the supplicants, like its GPS location or RSSI (Received Signal Strength Indication) or SNR (Signal to Noise Ratio).

3.5.2 Classification Based on Roles Played by Nodes

Regarding the existing key management protocols for mobile ad hoc networks, asymmetric cryptography is the main method for managing trust by using a public key infrastructure (PKI). Current PKI schemes are used in two different trust models: hierarchical and web-of-trust models.

Based on this classification, authentication protocols are divided into two groups: homogeneous and heterogeneous authentication schemes. Fig. 3.2 shows the classification of mobile ad hoc networks based on the functionality of the nodes in the network.

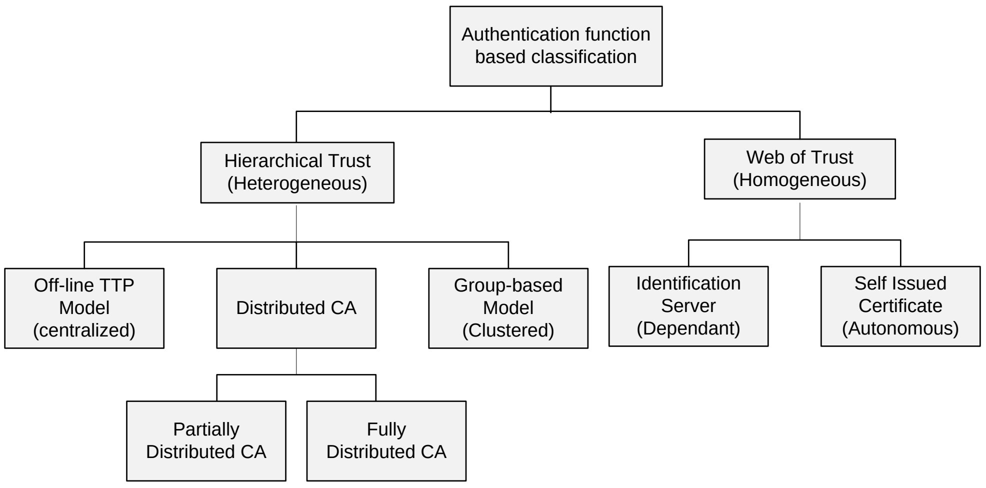

Authentication function based classification
Hierarchical Trust (Heterogeneous)
Web of Trust (Homogeneous)
Off-line TTP Model (centralized)
Distributed CA
Group-based Model (Clustered)
Identification Server (Dependant)
Self Issued Certificate (Autonomous)
Partially Distributed CA
Fully Distributed CA

Fig. 3.2. MANETs' authentication classification based on a node's role

3.5.2.1 Hierarchical Trust Model (Heterogonous)

In this class of authentication systems nodes in the network have different roles with respect to the authentication process. A central certification authority (CA) is considered as trusted third party (TTP) that is a source of trust for other nodes. This trust model is more structured because of the PKI and a fully trusted certificate authority that issues, verifies and revokes the certificates. In public key cryptography, CA has a public key, Pk, which is distributed to all nodes in the network and a private key, Pr. The CA stores the public keys of all nodes in the network. When a node requests to setup a secure communication to another node, the CA issues a certificate, signs it with its private key and distributes it to the requester node. The correctness of the certificate can be verified using the CA's public key [40]. Due to the dynamic nature of mobile ad hoc networks, applying TTP in form of a fixed CA is not considered.

There are authentication schemes in which CA distributes trust in a hierarchical manner. This scheme consists of a root CA located on the highest level; several delegated CA on a lower level and end users (Fig. 3.3). The root CA has the privilege to issue certificates to delegated CA and/or end users.

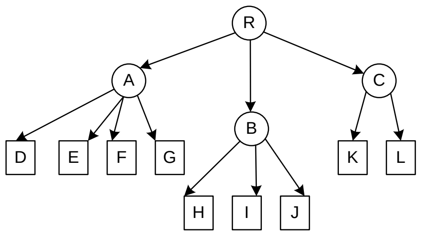


Fig. 3.3. Hierarchical Trust Model

Delegated CAs represent particular organizations, particular organizational units (like departments, groups or sections) or special geographical areas. An example of this infrastructure authentication scheme is PKI X.509 [41]. In these authentication schemes trust in a public key is achieved via a certification path. In order to get reliable knowledge of a public key, a user of a security service needs to obtain and validate a certificate. If an end user node does not have assured information (public key) of the CA that signed the certificate, then an additional certificate is needed to obtain the public key of this signer. In general, to obtain the reliable public key of an entity in the network, a chain of multiple certificates may be needed. The certificate chain or certificate path is a list of certificates used to authenticate a node in the network. Every certificate in the chain is signed by the node identified by the next certificate in the path.

Hierarchical trust models in ad hoc networks are classified into:

- Off-line centralized trusted third party: a key management task is mainly achieved by a trusted outside CA.
- Distributed certificate authority models (Threshold Scheme): key management is distributed among the set of CA.

3.5.2.1.1 Off-line Centralized Trusted Third Party

A hierarchical trust model was introduced by [42], where authentication performed locally, but an off-line TTP managed all certification tasks, like certificate issuing and revocation processes. The model is prone to a single point of attack.

3.5.2.1.2 Distributed Certificate Authority Model

This solution was first proposed in [43]; it allows the functionality of the certificate authority shared between a set of nodes in the network. This solution was designed to rescue the network in case of occurrence of a single point of attack by a compromised trusted third party. In this scheme the CA's public key is known by all nodes and the CA's private key, k, is divided and shared by n nodes. Fig.3.4 shows a distributed CA system.

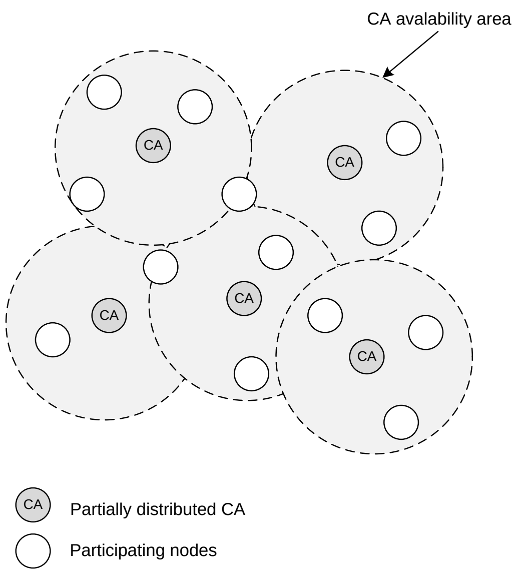


Fig.3.4. Distributed central authority [37]

Each CA signs certificates by its partial signature. The CA's signature is successfully created when a pre-defined number (e.g., *m*) of correct partial signatures is combined in a node. A secure authenticating process is possible when *m* CAs are accessible simultaneously. To share authentication tasks between several nodes threshold cryptography is used [44]. In this scheme *n* nodes are collaboratively capable of an authentication task by sharing the CA's private key; *m-out-of-n* node, *(m, n)*, can successfully perform the CA's task.

The private key, k, which is vital for a digital signature, is split into *n* parts $(S_1, S_{2,} \dots, S_n)$; this is represented in Fig. 3.5. Each part will be assigned to an authority node. Every authority node has its own public key, K_i, and a private key, $S_{i \cdot}$. In order to set up a secure communication with node A, its public key, K_A, is requested from the nearest authority nodes; therefore, CA's availability will be increased.

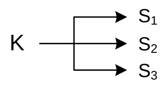


Fig. 3.5. Example of splitting a private key (2,3)

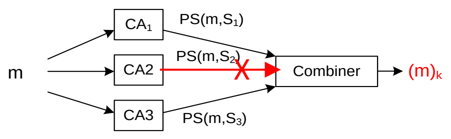


Fig. 3.6. (2,3) Threshold signature

Each CA, using its respective private key, provides a partial digital signature. A combiner node, which can be any node in the network, combines the partial signatures to create a signature for the certificate. *m* partial signatures are needed for issuing a successful certificate.

Fig. 3.6 shows a (2, 3) threshold scheme. In case of corrupting one CA, two other partial signatures are accepted when they meet the threshold requirements and are combined successfully in the combiner node to create the signature of the certificate. Although threshold cryptography increases the availability and security of a network by decentralizing the CA, there is still the possibility that a malicious attacker would compromise all CA authority nodes in the network. One counter-measure is proactive threshold cryptography [45]. It allows the shareholders, CAs, to periodically compute new key shares. The original private key does not change; only the partial shares held by each CA are refreshed. The MOCA scheme [46], Mobile Certified Authority, proposes a distributed certification authority solution for public key use threshold cryptography.

3.5.2.1.3 Cluster-Based Certificate Authority Model

This model is a hierarchical distributive key management system. It is appropriate for planned, long-term ad hoc networks with low capacity nodes which have resource limitation to perform public key management. These nodes are called pebble in [47]. The authors propose a cluster- or group-based symmetric authentication. As authentication is complex and costly for nodes with low capacity, authentication for clusters can minimize communication costs and increase efficiency. In this authentication scheme the lifetime of a network is divided into three phases: cluster generation, key update phase and the operation phase. In order to provide message-

and group confidentiality and authentication, the following keying materials are used [47]:

- Group key, k_G*:* is set in an initial setup phase of the network with an off-line authority and will be constant for the duration of the network. K_G is used to generate additional keys for security provision.
- Traffic encryption key, k_{TEK}: is used for symmetric data encryption and is updated during the network's lifetime.
- Cluster key, k_C: is generated by each cluster head and used for cluster specific communications.
- Backbone key, k_B*:* is used for secure communication between cluster heads.
- Hello key, k_H*:* is used between neighbors in the cluster generation phase.

In the *cluster generation* phase a cluster head and cluster members are specified. The appointment of a cluster head is based on the weight of the candidate node. This weight represents the current capacity of the node, like battery power, distance from other members etc. Each cluster head has the responsibility for managing group keying services for its cluster. In the *operation phase*, in each cluster nodes use k_G to authenticate other group members and provide message integrity. k_{TEK} is used to provide message confidentiality. In the *key update phase* a traffic encryption key, k_{TEK}, is periodically updated. k_{TEK} is generated by the *key manager* which is the most weighted cluster head among all cluster heads. The new traffic key is distributed to all other cluster heads in a secure way by encrypting it with a backbone key. Once a cluster head receives the new k_{TEK}, it distributes it to all cluster members, encrypted by a cluster key. The three phases are repeated in each network time segment. The shorter this time, the higher the security; however, the more required resources. Fig. 3.7 shows a cluster-based network.

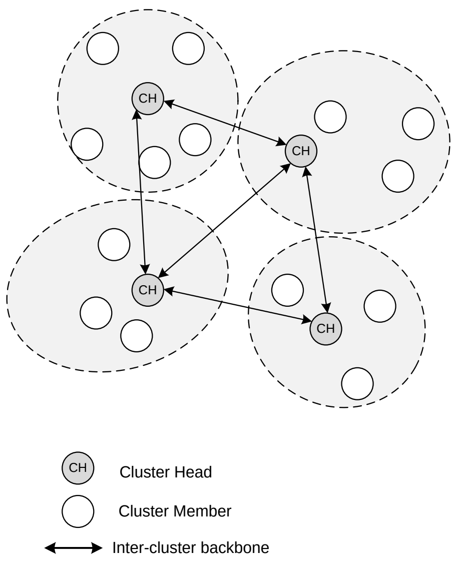


Fig. 3.7. Cluster-based network with cluster backbone [37]

Although the clustering approaches are suitable for large ad hoc networks by reducing security computation and complexity, a cluster head plays a role as a central authority and could be a single point of failure.

3.5.2.2 Web of Trust Model (Homogeneous)

Homogeneity (self-organizing) shows that all nodes in a network have the same role with regard to authentication processes.

In a web of trust model there is no particular certificate authority. Every node can act as a CA and provide its own trust opinion about other nodes (Fig. 3.8). In this class of authentication protocols each node in the network is the "center of its own world".

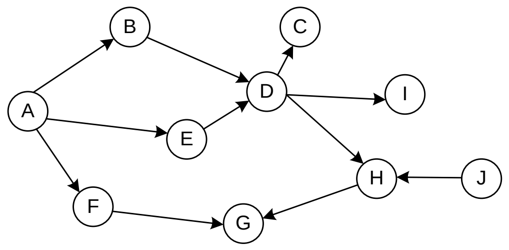


Fig. 3.8. Web of trust model

These authentication schemes are autonomous and every individual node is responsible for certificate management. Authentication mechanisms based on web of trust models are divided into two different classes: centralized web of trust; self-organized authentication mechanism.

3.5.2.2.1 Centralized Web of Trust

The Pretty Good Privacy model (PGP) [48], is an example for a web of trust model. In this model certificates are issued by the nodes themselves and may contain many digital signatures of other nodes. In hierarchical trust in the context of PKI, a user's identity is validated via certificates originated only by a CA; but trust in PGP can be achieved by finding a certificate path up to a trusted CA. Although PGP is a distributed approach to build a web of trust between individuals, as in PGP certificates are stored in an online centralized trusted repository, it is unsuitable for application in such dynamic and autonomous ad hoc environments.

3.5.2.2.2 Proximity-Based Identification

In this model authentication is based on demonstrative authentication [49]. In this solution a trust relationship can be established without prior knowledge and without existence of any off-line TTP. The model uses proximity channels for the bootstrapping phase and supplies a basis for further complex processes due to key establishment and key management.

In this approach, when two nodes wish to establish a secure communication link they need to perform two phases: pre-authentication and authentication. In the first step, the pre-authentication phase, nodes engage across a location-limited channel. This is a channel separated from the main communication channel used for exchanging pre-authentication information. Location-limited channels may be: physical contact, audio, infra-red etc. As an example, two PDAs[1] can direct their infra-red devices towards each other and exchange information. Regarding the nature of the infra-red communication, nodes can be assured that the received information stems from the chosen PDAs.

In order to establish the keying materials for secure communication purposes, after exchanging the pre-authentication information a key exchange scheme will be performed across the main communication channel. In the following the steps of key exchange for secure communication between nodes *i* and *j* are explained:

- Pair *i* and *j* makes proximity contact over a location limited channel.
- Node *i* sends the irreversible one-way hash function of its public key, $h(PK_i)$, to j and j sends $h(PK_j)$ to i. In this step pre-authentication information is exchanged.
- In their pair, i and j now exchange their public keys, PK_I and PK_J, over the main communication channel.
- To avoid an impersonation attack, authentication is proved by checking that $h(PK_I) = h(PK_i)$ and $h\left(PK_J\right) = h(PK_j)$.
- Upon successful verification any asymmetric key exchange protocol can be used to let pair i *and* j share a secret key and to establish secure communication.

A location-limited channel enables fully self-organized secure communication without the existence of any prior knowledge provided by an off-line trusted third party. This approach is suitable for spontaneous, small, localized short term ad hoc networks; for example, when people are gathering in a conference or a coffee shop where they want to establish a temporary communication network [37].

3.5.2.2.3 Self-organized Authentication Mechanism

Authors in [50] and [16] propose a fully self-organized authentication schemes adapted to the nature of ad hoc networks. In [51] the authors extend a demonstrative identification approach in a PGP-based network applied in large to moderate ad hoc networks. This authentication model is based on a self-issued certificate chaining approach. In Capkun's authentication model initial trust is established using location-limited communication and this trust is distributed network-wide relying upon mobility.

In contrast to PGP, where all certificates are stored in centralized certificate repositories, certificates in autonomous authentication systems are stored and distributed by the nodes. The system allows nodes to generate their key pairs, to issue certificates and to execute authentication processes. An authentication process is performed without the presence of a CA and there is no need of TTP organizational works even in the system initialization phase. Therefore, this solution is best suited for

[1] - Personal Digital Assistant

ad hoc environments; however, due to the complex initialization phase it is not suitable for small, short-term networks [37].

In general, a self-organized public key management system involves the following processes:

- Public/private key generation
- Certificate issuing and exchange
- Authentication
- Certificate revocation

The details of Capkun's self-organized authentication scheme are as follows [52]:

a. **Public/private key issuing and certificate creation**: in the initialization phase nodes in the network create their own public/private key pairs. Then nodes locally issue public key certificates based on the knowledge of the other public keys. A public key certificate is a data structure in which a public key is bound to an identity and is signed by the issuer of the certificate. A trust value corresponded to the issued certificate are not identifies but assumed to be issued, e.g., through a physical side channel.

b. **Certificate exchange**: certificates are exchanged periodically in $1/T_E$ frequency where T_E is the exchange period. Certificate exchanges are performed only between one-hop neighbors. Each node has two certificate repositories, each of them is modeled as a graph: an update certificate repository, G_u, and a non-updated, certificate repository, G_u^N. Once a certificate is issued, it is stored twice, in both the issuer's repository and the owner's repository. Each node periodically asks its physical neighbors, therefore, in the initial stage each certificate repository has only the certificates which it has issued and those which have been issued to it. Each node multicasts both G_u and G_u^N to its neighbors. In certificate exchange it is not the actual certificates that are sent, but rather unique identifiers (e.g., certificates' hash values) are transmitted. Then each node cross-checks the received sub-graph and its own sub-graph for additions and requests those certificates which it does not hold. After an initial convergence time, T_{CE}, each node will contain the whole certificate graph of the network. T_{CE} depends on the network properties. Only when a node encounters storage limitation for storing additional certificates, it removes expired certificates in order of their expiration time.

c. **Updated certificate repositories construction**: in the certificate exchange phase nodes obtain an incomplete view of the certificate graph and create their non-update certificate repositories. Therefore, nodes need a mechanism in order to build their update certificate repositories. The updated repository, G_u, is a sub-graph consisting of certificates which will be kept updated. The updated local certificate repositories can be constructed in two ways: a) either by exchanging the certificate graph with neighbors; or b) by applying the repository construction algorithm (A algorithm) to G_u^N which results in an updated certificate repository. The algorithm selects a sub-graph consisting of two distinct paths. During the execution of the algorithm the validity of the certificate is checked by contacting its issuers.

d. **Authentication**: after the initialization phase authentication will be performed as follows: when node S wants to authenticate the public key of node D, PK_D, it initiates the authentication process. S makes a request to D and asks for the list of hashes of updated certificate repository of D. After D replies with the list, S asks only for the certificates which it needs in order to complete the authentication by constructing the shortest certificate chain to PK_D in $G_S \cup G_D$. When a certificate path is found from S to D in G_S and G_D, the path will be investigated for validity and correctness of all certificates through the path.

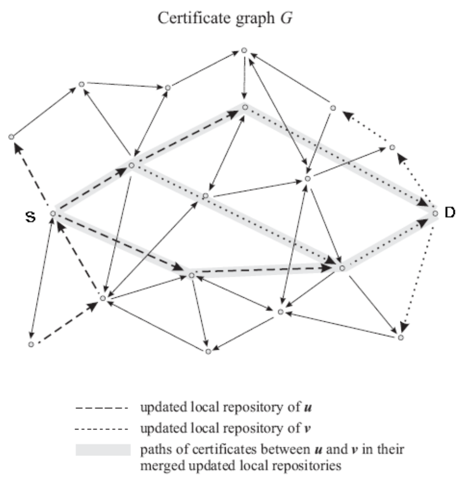

Certificate graph G
S
D
updated local repository of u
updated local repository of v
paths of certificates between u and v in their merged updated local repositories

Fig. 3.9. A certificate graph and paths of certificates between S and D in their merged, updated local repositories. [52]

e. **Certificate revocation:** a certificate will be revoked by the issuer if it believes that the user-key binding is no longer valid. There are different cases in which a certificate will be revoked:

- Implicit revocation occurs when a certificate has expired. Certificates are only valid for a pre-defined duration of time, T_v, which is defined flexibly. After expiration of this time a certificate must be updated. The model assumes that in this period a node is able to establish communication with any certificate issuer in order to exchange the updated version of the certificate.
- Explicit revocation happens when an issuer of a certificate believes that the ID-key binding of the certificate is not valid; or when its own private key is compromised.

The certificate exchange scheme lets the other nodes become aware of any certificate revocation at the time delay of T_{CE}.

f. **Coping with false certificates:**

- If node U receives a certificate which does not exist in G_u^N or G_u, then the certificate and its issuer will be labeled *unspecified* for a period of time T_P. After this period of time, if no conflicting certificate is received, the certificate will be changed to *non-conflicting*. In order to detect any inconsistent certificates for all nodes in a network it is necessary that $T_P > T_{CE}$.

- If a certificate conflict is found where a user U has received two certificate bindings for node V, (V, PK_V) and $(V, PK_V^{'})$, both certificates and the certificates which certified them are labeled *conflicting*. To resolve the conflict, node U first tries to check the validity of certificates contacting their issuers. If the certificates are still valid, U tries to find non-conflicting valid certificate chains to PK_V and $PK_V^{'}$. Receiving different certificate paths, U calculates the confidence value of each path corresponding to PK_V and $PK_V^{'}$, based on number and length of the chains. Two confidence values that show the U's confidence in the correctness of the two bindings are compared. In case of having convincing results, one of the user-key binding will be labeled *non-conflicting* and the other *false*. If no decision can be made, both bindings will be labeled *conflicting* and U will wait for gathering more information to resolve the conflict. In this work the conflict algorithm is assumed and not identified. In addition, the proposed conflict resolution mechanism can be used to evaluate the trust in users, to issue correct certificates and to detect malicious users.

The main problem of this scheme is the maintenance of two kinds of *updated* and *non-updated* certificate repositories. Its disadvantage is large overhead for storing the approximate global certificate graph in every node. To solve the problem and reduce the overhead, authors in [17] propose an on-demand localized public key management (*LPM*) solution. It provides a method to combine the certificate chain and the communication path. A certificate chain can be obtained hop-by-hop, as long as a route is discovered between source node and destination node; subsequently an authentication process can be achieved. In this scheme all certificates need to be issued and trusted locally. As a result, misbehaving can be detected easily in the neighborhood. Additionally, a certificate update is performed in only one-hope distance – unlike Capkun's scheme in which many hops are involved when updating certificates. Furthermore, revocation is performed in one- or two-hop distance. Hence, a reduction of costs can be achieved by the recent work when compared with the scheme in which the revoked certificate list will be informed to the nodes in the system. In the LPM scheme the average consumed capacity for maintaining repository is less than in Capkun's scheme. Therefore, the proposed on-demand scheme can achieve good scalability and availability and is more appropriate for the self-organized nature of ad hoc networks. However, it lacks a mechanism that can cope with malicious behavior. [18] proposed a solution based on an existing web of trust in integration with the AODV routing protocol. For the authentication process, a discovery of two independent certificate chains is required. They propose a solution to cope with the malicious nodes. However, they assume the existence of a web of trust.

3.6 Thesis Authentication Mechanism

The authentication mechanisms proposed in this thesis is based on a self-organized authentication mechanism without relying on any central authority and any pre-defined web of trust [53][54] The scheme allows nodes to create their own public-private key pair and issue certificates for neighboring nodes in the network. Certificates are locally stored on each node and are distributed by them in the network. The trust model of our scheme is based on creating a web of trust of public key certificates that guarantees bindings of the public keys to their related user identities. Trust value, or node's belief, represents the confidence value of the trustworthiness of another node.

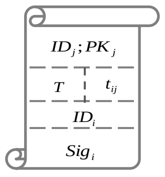


Fig. 3.10. Public key certificate of node *j* issued by node *i* ($Cert_{i\rightarrow j}$)

Fig. **3.10** shows the structure of a public key certificate in a trust-based, self-organized scheme. It is composed of the node's identity and the corresponding public key which are bound together; the identity of the certificate issuer; the trust value; duration of the validity time. Issuer of the certificate signs the certificate with its private key and adds the signature to the certificate for certificate verification purpose.

In this self-organized authentication mechanism, public key authentication is performed through a chain of certificates. As an example, $Cert_{i\rightarrow j}$ denotes the certificate that *i* issued for *j* and signed with its private key, Sig_i, to show the binding of node *j*'s identity, ID_j, and its corresponding public key PK_j. In addition, certificate $Cert_{i\rightarrow j}$ should contain the identity (network address) of issuer, ID_i, certificate validity time, T, and trust value or confidentiality, t_{ij}, representing the level of issuer's assurance of the binding of ID_j to its corresponding public key, PK_j. A web of trust can be considered as a certificate direct graph *G(V, E)*, whose set of vertices, *V*, represents the public keys and the set of edges, *E*, represents the certificates.

We assume each node, depending on the expiration time of certificates, creates periodically direct edges to its neighbors and issues certificates for them if there is an acceptable confidentiality level for binding the neighbors' *ID* to their corresponding *PK*.

When a node, *S*, wants to authenticate the public key of another node, *D*, which is not located in radio range of *S*, a chain of valid certificates from *S* to *D* is required, as seen in Fig. 3.11.

$$\begin{tikzcd} S \arrow[r, "Cert_{S\rightarrow B}"] & B \arrow[r, "Cert_{B\rightarrow C}"] & C \arrow[r, "Cert_{C\rightarrow D}"] & D \end{tikzcd}$$

Fig. 3.11. PK Certificate chain

In our example the certificate chain from S to D is $\left\{Cert_{S\to B}, Cert_{B\to C}, Cert_{C\to D}\right\}$. The first certificate, issued by node S, will be verified by its PK_s. Every certificate in the chain will be verified with the public key of the previous certificate in the chain. For example, in Fig. 3.11 the correctness of the certificate issued by node B for node C will be verified by B's public key; subsequently, the certificate issued by node C for node D will be verified by the public key of C.

3.6.1 Trust Metrics

Trust metrics is a measure that represents the assurance that a requesting node can obtain the public key of a destination node correctly through the certificate chain. In this chain fashion, trust transitivity plays a big role which is based on recommendations between entities. Referring to the unidirectional characteristic of a trust relationship, the same assurance in the reverse direction of the certificate chain, at the same time, need not exist.

A variety of trust value representations and measuring is proposed, e.g., discrete trust values as in PGP or continuous trust values. In our trust model we consider trust to be a continuous value in the range of [0, 1]. However, there is a difference between trusting an entity to provide a specific service and trusting an entity to recommend someone who can provide the service [55]. Trust in the service object is functional trust, while trust in recommending agents is referral trust. In our model we consider the functional trust as the honest binding rate, i.e., the number of correct binding signs over all trials. On the other hand, referral trust is the dissemination of these scores to the relying nodes that can be considered as recommendations.

Any node in the network can calculate the trust value of another node's public key if there is a physical communication and, consequently, a certificate chain between the two nodes using formula (3.1) [18].

$$t_{SD} = \prod_{k=1}^{n} t_k \tag{3.1}$$

t_k is the trust value between two directly connected nodes on the certificate chain from node S to node D. n is the number of hops between the source and the destination. It is obvious that the trust in another's public key fades along the path of recommendation. The final trust value of the certificate chain will not be larger than the lowest trust value and the final confidence value of the certificate chain is as strong as the weakest path.

3.6.2 Security Threats

The lack of any infrastructure poses the greatest threat to the establishment of secure key management systems. There is no guarantee in self-organized public key management systems that all nodes will act correctly and honestly. In general, two types of misbehavior threaten the security of our trust-based authentication model:

- *functional* misbehavior
- *referral* misbehavior

- **Functional misbehavior:** *functional misbehavior* occurs when a node, or a group of nodes, refuses to act correctly in service provision. In our authentication model we assume this can take place by not participating in the authentication process or by *impersonating* another node. By impersonating another node a malicious node, B, may issue a certificate that binds another node‘s identity, ID_C, to its public key, PK_B, and signs it with its private key Pr_B, in the signature form of SigB (Fig. 3.12). The aim of the malicious node is to eavesdrop messages sent to C.

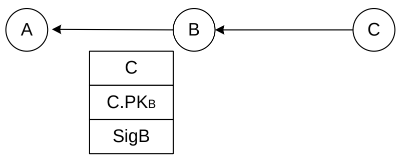


Fig. 3.12. Node B impersonates node C in order to eavesdrop the messages from A to C

In this work we assume that all nodes voluntarily participate in the authentication process. In addition, it is assumed that nodes do not delete the certificates from a certificate chain with the aim of preventing the certificate chain discovery phase.

- **Referral misbehavior:** *referral misbehavior* happens when a malicious node inserts a false certificate in a certificate chain or makes other nodes believe in a false certificate.

  - Inserting a false certificate in the chain results in a failure in the certificate chain verification process. This can be performed by binding a public key of node C, PK_c , to another node like D, ID_D, although it should be bound to ID_c. Certificate verification has failed if the signature of the certificate cannot be decrypted by its corresponding public key; furthermore, it can fail when the origin of the certificate is not the one who signs the certificate.

  - A malicious node tries to trick other nodes by providing dishonest recommendations, i.e., by manipulating the confidence in the authenticity of a given key. One important threat of this kind is the Sybil attack [21], where a malicious node generates several keys and identities, binds the IDs to corresponding public keys and issues certificates for them. In this case the malicious node can use these nodes to issue false certificates. As the false certificate is signed by many Sybil nodes, it could be considered a correct certificate to non-Sybil nodes (Fig. 3.13)

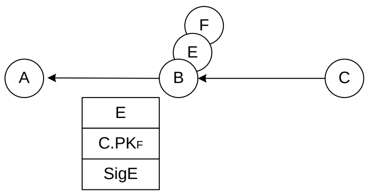


Fig. 3.13. Sybil Node B generates multiple IDs and corresponding key pairs in order to issue and sign faked and dishonest certificates.

A Sybil node tries to deceive other nodes so as to make them believe in a false certificate.

In this thesis we concentrate on misbehaving nodes that try to harm the authentication service and aim at disseminating false information by inserting false certificates into a certificate chain.

When it is not in our power to determine what is true,
then we out to follow what is most probable.
- Rene Descartes

Chapter 4
Bio-inspired Learning Mechanism

Traditionally, in the realm of approaches for data routing, protocols and applications have been designed to work in a hierarchically organized static network in which address are assigned to nodes. In order to find a path in the network every node maintains a static routing table. Any topology change in the network should be propagated through the whole network. Therefore, latency and loss of data increase until new updates in routing tables is successfully done. Furthermore, considering the *security* issues with regard to the characteristics of wireless mobile ad hoc scenarios, traditional protocols cannot be successfully applied. The non-existence of infrastructure, mobility, increase of node failures and different malicious behaviors affect the cost of data transmission and quality of services. Moreover, keeping global information fresh is energy consuming and is not suitable for an energy-restricted environment. In order to overcome the given problems, the domain of machine learning attracts researchers' attention. The aim of machine learning algorithms is an automatic learning of the environment's properties and adapting their behavior to the environment easily and quickly.

A survey of machine learning techniques, applied to wireless ad hoc networks, is presented in [56]. It gives a short description of suitable machine learning approaches for use in wireless ad hoc networks, based on the most relevant requirements of these networks' environment. Different approaches, like reinforcement learning [57][58], swarm intelligence [59], mobile agents [60] and real time heuristic search [61][62] [63] are proposed. Further approaches such as genetic algorithms [64], neural networks [65], decision trees and rule learners [66] are proposed where global information is available or needed, e.g., optimal sensor node placement. All these approaches are evaluated in terms of their quality – meeting requirements of the application. Existing machine learning mechanisms for routing in ad hoc networks are reinforcement learning, swarm intelligence or mobile agents.

Each node in a MANET is an autonomous computational unit that communicates with its neighbors. Because a central authority and any kind of infrastructure are absent, and also because of frequent topology changes, algorithms require global information from the whole network; this is very expensive. Therefore, data is distributed through the nodes of the network. In distributed machine learning algorithms, each node holds only part of the information about the relevant problem (e.g., the best next hop in a routing problem) and the complete solution is found in collaboration.

A summary of the proposed distributed machine learning approaches with respect to their properties which are relevant to wireless ad hoc scenarios is shown in Table 4.1.

Table 4.1. Characteristics of Machine learning techniques for ad hoc scenarios [56]

Machine learning technique	Memory requirements	Computation requirements	Flexibility to topology change	Optimal results	Initialization cost	Overhead
Reinforcement learning	medium	medium	high	high	medium/high	low
Swarm intelligence	medium	medium	high	high	high	medium
Heuristic	medium	low	medium	medium	high	low
Mobile agents	low	low	high	high	low	medium

Considering the infrastructure-less and mobile nature of wireless ad hoc networks, and considering also no energy limitation, compared to wireless sensor networks, swarm intelligence seems to be the best choice out of the machine learning algorithms to be applied in MANETs. Since in such networks individual nodes are mostly laptops and PDAs, which can be easily rechargeable, energy is not restricted. In addition, the computation requirements are not high and their flexibility with respect to topology change is high. No infrastructure and the self-organizational characteristic of the ad hoc networks cause high costs for the initialization phase. These costs are due to the initial learning process in which the network learns to deal with the uncertain environment. However, the probability of finding the optimal result is high with swarm intelligent algorithms.

4.1 Swarm Intelligence

Swarm intelligence refers to a class of machine learning techniques, inspired by natural biological systems such as colonies of ants, flocks of birds and shoals of fish. Swarm intelligence algorithms are made up of individuals, called agents, which locally and, to a certain degree randomly, follow simple rules with no centralized control that tells them how they should behave. These unsophisticated agents cooperate through self-organization by interacting locally with their environment. Collective behaviors and interactions between such agents lead to the emergence of an "intelligent" global behavior, unknown to individual agents.

Fig. 4.1 [1] . Colonies of ants

Fig. 4.2 [2]. Flock of birds

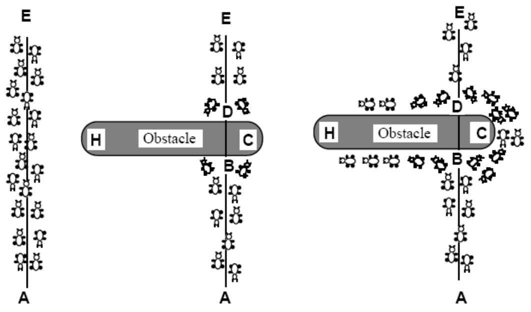
Fig. 4.3 [3]. Shoal of fish

Among these bio-inspired learning mechanisms the most studied and most successful is the general purpose optimization technique based on ant colony optimization [67]. Ant colony optimization (ACO) is inspired by the foraging behavior of real ants. Along their path between food source and nest, ants deposit pheromones on the path in order to make some favorable path to be followed by other members of the colony. ACO exploits a similar mechanism for solving discrete optimization problems where the aim is – in a distributed and cooperative manner – to find the *best* solution among all feasible solutions.

4.2 Ant Colony Optimization

4.2.1 Biological Inspiration

Biological inspiration of ant colony optimization is based on *stigmergy*. Stigmergy is a mechanism of indirect communication between agents. It describes a particular type of communication in which the workers are stimulated by the performance they achieve. An ant deposits a substance known as *pheromone* in the environment. Other ants perceive the presence of the pheromone trails on the path and tend to follow the paths with higher pheromone concentration.



a) b) c)

1 - http://www-lih.univ-lehavre.fr/~bertelle/antColonyCoop.jpg
2 - http://staff.washington.edu/paymana/swarm/swarm.jpg
3 - http://ih3.redbubble.net/work.2368403.3.flat,550x550,075,f.lightbulb-schooling-fish.jpg

Fig. 4.4. Example of real ants: a) ants follows a path between points A and E. b) an obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability. c) On the shorter path more pheromone is laid down.

In Fig 4.4.a ants traverse the path between nest E to the food source A and vice versa. Suddenly the path is broken because of an obstacle. Therefore ants in position D and B on the path have to decide whether to turn right, or left (Fig. 4.4.b). The decision is based on the intensity of the pheromone trails left by preceding ants. However, immediately after cutting off the path, there is no pheromone on the two alternative paths. In this situation, ants choose a path randomly. As the path BCD is shorter than the path BHD, ants will traverse it faster. Therefore, returning ants will find a stronger pheromone trail on this shorter path, DCB. As a consequence, the number of ants which traverse the shorter path will be higher than the number of ants which traverse the longer one. As a result, the amount of pheromone on the shorter path grows faster than on the longer path. The probability, therefore, that a single ant will choose a path to follow is quickly biased toward the shorter path.

4.2.2 The Optimization Technique

In the early nineties the first ant colony optimization (ACO), called Ant System, was introduced by Marco Dorigo [68]. ACO is a type of metaheuristic algorithm for combinatorial optimization [69]. A metaheuristic is the most general kind of algorithms and techniques that employ some degree of randomness to find optimal solutions to hard problems [70]. Metaheuristic algorithms are applied to problems for which a brute-force search is out of the question, because the space is too large, and to problems that are $NP -$ hard problems, where there is no algorithm for solving them in polynomial time. Dynamicity can make the search situation even harder. ACO algorithms are applied to problems for which it is not clear what an optimal solution could look like, or even how to go finding it. In this kind of algorithms some candidate solution is given to the problem as an initial phase. Then these solutions will be tested and assessed for how optimal they are. The new candidate solution will then be made by some random modification to the old candidate solution. If the new solution is better, the old one will be ignored; otherwise the new solution will be ignored. The process continues iteratively in order to improve the candidate solution until reaching the stop conditions.

Metaheuristic are general-purpose algorithms suitable for being applied to different optimization problems. In ACO each ant separately builds a solution. But the main component of ACO is cooperation. Although each individual ant can build a feasible solution, high quality solutions are the result of cooperation among all agents of the colony, who concurrently builds different solutions. The agents exchange information by affecting and modifying some aspects of their environment. This information has the role of a pheromone trail, by analogy to the real ants' life. An artificial pheromone trail is numerical information locally stored in the problem's states, visited by artificial ants. In ACO local pheromone traces are the only communication way between the ants in the colony. Pheromone traces, left by each individual ant in the environment, result in a perceived, collective knowledge about the environment. This stigmergy has a role as a function of the past history of the whole colony.

Similar to real pheromone evaporation, in ACO, an evaporation mechanism reduces pheromone values over time. This allows slowly forgetting past history and enables ants to search new directions instead of being over constrained[1] by past decisions. The evaporation mechanism avoids a fast convergence of all ants to the same part of the search space.

In ACO, the ant's movement is a transition from discrete states to discrete states. An amount of deposited pheromone[2] is a function of the quality of the solution. Moves into next neighbor state are based on the value of the pheromone trail and problem-specific local information.

Decisions about the time for releasing pheromone in the environment and the pheromone's amount depend on the characteristic of the problem and the implementation's design. Ants can release pheromone while building the solution; this is called *online step-by-step pheromone trial update;* or after a solution has been built, moving back to all visited states is called *online delayed pheromone trailed update*; or both [69]. Autocatalysis is an important characteristic of ACO algorithms. The more ants choose a path, the more the path is rewarded by increasing pheromone value and the more attractive path it becomes for the next ants. Once an ant accomplishes its task, it 'dies' and is deleted from the system.

In addition to the components acting from the local perspective, like ants generation, local search, pheromone update and evaporation, the ACO metaheuristic can also include some extra components which use global information. These components are named *daemon actions* in the algorithm. With a daemon observing the ant's behavior and collecting useful global information, depositing additional pheromone is allowed. This way an ant search process will be biased from a non-local perspective. One example is observing the quality (amount of pheromone which ants deposited online) of all solutions generated by the ants and deposit additional pheromone only on the components related to some of the solutions. This additional pheromone update is called *offline pheromone trailed update*.

4.2.3 Combinatorial Optimization

A combinatorial optimization is applied in problems for which exhaustive search is not feasible. It operates on those operation problems in which a solution consists of a combination of components selected from a finite set. The set of feasible solutions is discrete with the objective to find the optimal/best solution which is the combination of components.

[1] - Over constraint means to accidentally constrain the system to eliminate possibilities which are intended to allow.

[2] - In reality, some real ants deposit more pheromone in case of finding richer food source.

A model of combinatorial optimization problem [71], [19] consists of:

- Search space **S** defined over a finite set of discrete components/decision variables: $N = \{n_1, n_2, \dots, n_l\}$. Each member of S is presented as ordered component sequences, e.g., $\delta \epsilon S$; $\delta = < n_r, n_s, \dots, n_u, \dots >$
- A set of constraints $\mathbf{\Omega}$ defined for the problem under solution.
- An objective function $f: S \to R_0^+$ to be minimized / maximized. This function is associated with each solution and is also known as cost function, C (S).
- A set of all feasible sequences with respect to the constraints $\mathbf{\Omega}$.

In ACO the model of combinatorial optimization problem is used for defining the pheromone model.

Combinatorial optimization problems can be represented in the form of the weighted graph G(V, E), $w$here V is a set of vertexes and E is the set of edges that connects the set of components, e.g., e_{ij}. An artificial ant builds a solution by traversing a graph, vertex to vertex along the edges of the graph, building a partial solution. The graph G is called constructive graph and we have:

- The components n_i that are the vertices of the graph
- The edges of the graph, e_{ij}, are the connection between the neighborhood vertices. Transition is between the current vertex and the next accessible vertices in the graph.
- The states $\delta \epsilon S$ correspond to the paths in the graph, e.g., sequences of nodes or edges.
- A pheromone value, τ_{ij}, is associated with the components of each possible solution. Ants deposit a certain amount of pheromone on the components; that is either on the vertices or on the edges they traverse.

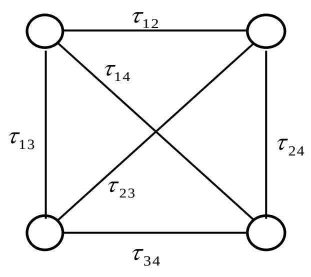


Fig.4.5 an example of weighted graph G$(\mathbf{V}, \mathbf{E})$

- A search process for choosing a solution component is guided by a stochastic mechanism, biased by the pheromone values associated with next components, τ_{ij}, and heuristic values, η_{ij}, which represent some heuristic information about the problem under solution.

4.2.4 Generic Structure of an ACO Algorithm

The structure of a generic ACO Algorithm is presented in algorithm 4.1 [69] [19]. It shows briefly the ACO metaheuristic. After setting the parameters and initializing the pheromone value, the algorithm iterates over three phases. For each iteration m number of ants is generated and constructs solutions. In each construction step a partial solution is extended by adding a feasible solution component with regard to the constraints. Creation of a solution can be regarded as a walk on the graph G(V, E).

The pheromone_update procedure refreshes the pheromone values and daemon action is optional.

Algorithm 4.1 The Ant Colony Optimization Metaheuristic

```
Procedure ACO_metaheuristic()
    set_ parameters, initialize_ pheromone_trails
    while (termination_criterion_not_satisfied)
        schedule_activities
            ants_ generation_and_activity();
            pheromone_update();
            daemon_actions(); {optional}
        end schedule activities
    end while
end procedure
```

Algorithm 4.1. ACO Metaheuristic

The process continues until the maximum number of iterations is reached or all ants find the same solution. This situation is called *stagnation behavior*. It indicates a situation in which the algorithm stops searching for alternative solutions, because in the searching phase all ants are attracted to the best solution, i.e., the one which has the highest pheromone trails.
Algorithms 4.2 and 4.3 represent the generation of ants and their activities respectively. Procedure update_ant_memory initials the state from which the ant starts its path.
The procedures compute_transition_probabilities and apply_ant_decision_policy consider the current state of the ant, the current values of the pheromones and heuristic information to establish the probabilistic transition process to other feasible states.

Algorithm 4.2 Ant generation and its activities

```
Procedure ants_ generation_ and_ activity()
    repeat in parallel for k=1 to m (number_of_ants)
        new_ant (k);
    end repeat in parallel
end procedure
```

Algorithm 4.3 Activities of a new ant

```
Procedure new_active_ant()  {ant lifecycle}
    Initialize_ant();
    M = update_ant_memory();
    while (current state $\neq$ target state)
        A = real_ local_ant-routing_ table();
```

```
            P = compute_ transition_ probabilities(A, M, problem_ constraints);
            next_state = apply_ ant_ decision_ policy(P, problem_ constraints);
            move_ to_ next_ state(next_state);
            if (online_step-by-step_ pheromone _update)
                deposit_ pheromone_on_the_visited_arc();
                update_ ant-routing_ table();
                daemon actions(); {optional}
            end if
            M = update_ internal_ state();
        end while
        if (online_delayed_pheromone_update)
            evaluate_ solution();
            deposit_ pheromone_ on_ all_visited_arcs();
            update_ant-routing_table();
        end if
        die();
    end procedure
```

Algorithm 4.2 and 4.3. ACO Metaheuristic

4.3 ACO Algorithms

The first ant colony optimization, called Ant System (AS), was introduced in [72]. Since then, other ACO algorithms have been introduced. Table 4.2 shows a list of proposed ACO algorithms [67].

For many applications, in a construction graph G(V, E), pheromone values and heuristic information are assigned to the edges.

Table 4.2. Proposed ACO algorithms

Algorithm	*Authors*	*Year*	*References*
Ant System (AS)	Dorigo et al.	1991	[73], [68], [72]
Ant-Q	Gambardella & Dorigo	1995	[74]
Ant Colony System	Dorigo & Gambardella	1996	[75], [76], [77]
MAX–MIN AS	Stützle & Hoos	1996	[78]
Rank-based AS	Bullnheimer et al.	1997	[79]
ANTS	Maniezzo	1999	[80]
BWAS	Cord´on et al.	2000	[81]
Hyper-cube AS	Blum et al.	2001	[82]

In the following the most applied ACO algorithms are presented.

4.3.1 Ant System (AS)

The importance of the AS algorithm [72] lies with it being the prototype of a number of ant algorithms. The algorithm is executed during t_{max} iterations. In each iteration m ants start to build the solutions. By moving from one vertex/node, i, to another node, j, the edge (i,j) is added to the solution under construction. Three variants of the AS algorithm have been proposed: *ant-density*, *ant-quantity* and *ant-cycle*. These algorithms differ in the way the pheromone values are updated. In the first two algorithms ants deposit pheromone while they build a solution (*online step-by-step pheromone trial update*); in the latter ants deposit pheromone after building the complete tour (*online delayed pheromone trailed update*). Experiments [73], [68], [72] have shown that *ant-cycle* preforms much better than the other two algorithms. Consequently, *ant-cycle* became known as Ant System and the other two algorithms were discarded.

During the construction of solutions by ants, in each step an ant, a, located in node i chooses the next node j to move to, with the probability calculated as follows:

$$m_{ij} = \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k\epsilon N_a(i)}[\tau_{ik}]^{\alpha}.[\eta_{ik}]^{\beta}} \tag{4.1}$$

$$p_{ij}^{a} = \begin{cases} m_{ij} & \text{if } j \epsilon N_A(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where τ_{ij} is the pheromone deposit on the edge between vertices i and j; η_{ij} is the heuristic value of moving from node i to node j, given by:

$$\eta_{ij} = {}^{1}\!/_{d_{ij}} \tag{4.3}$$

where d_{ij} is the distance between i and j. In (4.1), $N_a(i)$ is the list of feasible neighboring nodes of ant a when located at node i, and α and β are the weights for balancing the deposited pheromone and the heuristic value of the edge, respectively. Each ant has a memory that keeps the nodes visited already. Using this memory, in each step an ant moves to an unvisited neighbor node. α shows the relative importance of the pheromone trail and β represents the relative importance of the heuristic value.

In [73], a trade-off between the effect of pheromone values and heuristic information was investigated by varying α and β. If $\alpha = 0$, pheromone values are no longer considered, with the result of an algorithm close to *stochastic greedy algorithm* (with multiple starting points if m ants are initially located in different nodes) is obtained. If $\beta = 0$, only the pheromone values are considered for guiding the constructive process. This can cause a quick stagnation, where ants always construct the same solution by leading to the pheromone trails related to some transitions.

An investigation of the *ant-cycle* algorithm with regard to different combinations of α and β shows that choosing a high value for α enters the stagnation behavior very quickly without finding the optimum solution; setting α to a low value, the algorithm cannot find a good solution either. Therefore, a proper balance between the importance of pheromone value and heuristic is needed.
When all ants complete constructing their solutions, pheromone trails of all edges visited by ants will be increased:

$$\tau_{ij} = (1 - e).\tau_{ij} + \Delta\tau_{ij} \tag{4.4}$$

where $e \epsilon (0,1]$ is a coefficient such that $(1 - e)$ is the rate of pheromone evaporation. Pheromone evaporation is a function of time and allows the system to forget the old information, search new paths and also avoid convergence to premature-optimal solutions by encouraging exploration of edges not yet visited. The main role of pheromone evaporation is to prevent a quick stagnation situation in which all ants converge to the same solution. $\Delta\tau_{ij}$ is the amount of pheromone deposited with typically the following form:

$$\Delta t_{ij} = \sum_{k=1}^{m} \Delta t_{ij}^{k} \tag{4.5}$$

where Δt_{ij}^{k} is the pheromone that ant *k* deposits on the edge (i, j) that it has passed; this is given by:

$$\Delta t_{ij}^{k} = \begin{cases} Q/L_k & \text{if} \quad \text{ant } k \text{ used edge } (i,j) \text{ in its } solution \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where Q is a constant and L_k is the length of the solution constructed by ant *k*.
An extended version of AS, called *elitist AS*, is proposed in [72]. It shows better performance than the original AS. In elitist AS, the daemon action deposits additional pheromone on the edge belonging to the best solution. The amount of deposited pheromone depends on the quality of the global best solution:

$$\tau_{ij} = \tau_{ij} + a.\Delta t_{ij}^{best}, \ \forall edge(i,j) \epsilon S_{global-best} \ ; \Delta t_{ij}^{best} = f\left(C(S_{global-best})\right) \tag{4.7}$$

where *a* is the number of elitist ants.

4.3.2 Ant Colony System (ACS)

An ant colony system was proposed by [75], [76]and [77]) to improve the performance of AS. ACS is based on AS; however, there are important differences between them:

- ACS uses a different transition rule called *pseudo-random proportional rule.* It is used as follows:

$$p_{ij}^{k} = \begin{cases} 1 & \text{if } \ (q \leq q_0)\ and\ (j = \ argmax_{k \,\epsilon\, N_{a\,(i)}}\{t_{ik} \,.\eta_{ik}^{\beta}\}) \\ & m_{ij}\ \ if\ (q > q_0) \\ & 0\ \ \ otherwise \end{cases} \tag{4.8}$$

  where p_{ij}^{k} is the next chosen node by ant k in its next movement, q_0 is the probability of choosing deterministically the most promising edge, q is a measure in range of [0, 1]. Considering q_0, the decision rule provides two possibilities: When $q \leq q_0$, the available knowledge will be exploited. This means that the ant's next movement is toward the node which is the best option with respect to heuristic information and the learned knowledge memorized as the form of pheromone value. However, when $q > q_0$, a new connection will be explored. q_0 modulates the exploration activity, whether it is based on the best available solutions or on the search space.

- Pheromone update takes place only through daemon action (*offline pheromone update*). By the end of the building solutions stage, the pheromone values of edges that belong to the global best solution are updated. In ACS pheromone evaporation, like pheromone deposit, is only applied to the edges of the global best solution. The daemon action is considered as follows:

$$\tau_{ij} = \tau_{ij} + e.\Delta t_{ij}^{best},\ \ \forall edge(i,j)\epsilon S_{global-best} \tag{4.9}$$

$$\Delta t_{ij}^{best} = f\left(C(S_{global-best})\right) \tag{4.10}$$

- Ants in ACS perform only *online step-by-step pheromone updates* through building the solutions. These updates encourage the generation of other solutions rather than the best one, and help to avoid that every ant follows the same path.

$$\tau_{ij} = (1-\varphi).\tau_{ij} + \varphi\tau_0 \tag{4.11}$$

where $\varphi \in (0,1]$ is considered as the second pheromone decay parameter, τ_0 is the pheromone initial value[1] .

ACS's local update makes the visited edges less and less attractive as ants traverse them, and results in the exploration of not yet visited edges.

ACS is the successor of Ant-Q [74] which tried to merge AS and Q-learning properties [19]. The only difference between Ant-Q and ACS is the value of τ_0; however, both result in almost the same performance.

4.3.3 Max-Min Ant System (MMAS)

Max-Min Ant System [78] is the improved extension of AS. Similarly to ACS it applies an *offline pheromone trail update*. That is, only the best ant updates the pheromone values and the pheromone values are bounded:

$$\tau_{ij} = [(1-\rho).\tau_{ij} + \Delta t_{ij}^{best}\,]_{t_{min}}^{t_{max}} \quad ; \quad \forall edge(i,j) \epsilon S_{best} \tag{4.12}$$

where t_{max} and t_{min} are the upper and lower bounds respectively and Δt_{ij}^{best} is:

$$\Delta t_{ij}^{best} = f\big(C(S_{best})\big) \tag{4.13}$$

This may be either the best solution found in the current iteration, or the global best solution since starting the search process. Boundary pheromone values, t_{max} and t_{min}, are tuned to the problem under consideration. Some guidelines have been proposed for designing these values in [78].

4.4 Application of ACO

Referring to the concurrent and adaptive nature of ACO algorithms, they are mainly suitable for distributed stochastic problems where the behavior of the system is intrinsically non-deterministic. ACO is mainly applied to *NP*-hard problems in which the optimal solution can be found with exponential time worst case complexity. ACO was initially applied in Traveling Salesman Problem (TSP). TSP became a common test-bed for several ACO algorithms to achieve a better performance [76]; [78]; [83]. Other problems tackled by ACO are assignment problems where a set of items, like activities or objects, have to be assigned to a resources like agents or locations,

[1]- In three heuristic rules for sequencing jobs to a single production facility', experimentally it is shown that $\tau_0 = (nL_{nn})^{-1}$ produces good results for solving Traveling Salesman Problem. where n is the number of the cities and L_{nn} is the length of a tour produced by the nearest neighbor heuristic

considering some constraints. Quadratic assignment [80], course timetabling [84] and graph coloring [85] are examples of assignment problems. Another application is scheduling problems, which deals with the allocation of scarce resources to task over time. Project scheduling [86] and open shop [87] are examples for these problems solved by ACO. Author in [88] presents a classification of ACO applications.

Other applications include dynamic shortest path problems related to communication and routing in a telecommunication network. They are applied to wired and wireless networks. AntNet [89] is the application developed for a wired network. It is used to find the near-optimal route. Each node in the network has two data structures: a pheromone table which keeps, for each destination *d*, the pheromone value of all neighboring nodes to reach *d*; and a local traffic statistic table which stores the routing information as the cost of communication. The local traffic statistic/cost table stores estimated average, variance and best trip time for each destination. There are two kinds of *forward* and *backward* ants. In regular intervals every source node sends out forward ants to all known destinations. During their travel, forward ants record the times arriving at each node on their way. Once reaching a destination, the forward ant changes into backward ant, traverses the exact path of the forward ant and updates the pheromone values based on the gathered cost information.

One important disadvantage of AntNet is high communication overhead caused by regularly sending out ants.

In the realm of soft security mechanisms, several trust and reputation systems have been proposed in different domains – ranging from human social networks, e-commerce, peer-to-peer networks to mobile ad hoc and sensor networks. Each trust model is composed of two main components: a trust computation model and a trust evidence distribution system. The distribution part is the basis for the computation part.

AntRep [90] is another proposed *trust-reputation evidence distribution* scheme using an algorithm based on bio-inspired ant colony systems in P2P networks.

TACS [3] is a proposed trust model, based on an ant colony system, that provides guarantees of network resource availability and trustworthiness in P2P networks.

4.5 Existing ACO Approaches in Mobile Ad Hoc Networks

- Routing Domain

AntHocNet [91], [92] is an adaptive algorithm for shortest path discovery in mobile ad hoc networks. It is a hybrid, reactive and proactive routing protocol. In order to reduce the overhead, the routes will be discovered reactively. Small control packets, called *reactive forward ants*, gather routing information from source to destination, e.g., end-to-end delay or number of hops. After reaching a destination, forward ants change into backward ants and traverse the exact path of the forward ant back to the source node. On the way back, using collected information, a routing table will be updated at each intermediate node. Paths are set up based on the pheromone values indicating their quality. Based on the pheromone table data, packets are routed stochastically through different paths. However, to have better performance the protocol proactively monitors, maintains and improves the existing paths using *proactive forward ants.* This approach outperforms the AODV routing protocol on different evaluation criteria.

- Security Domain

In the domain of security, [36] proposed a model – ABED, *Ant-Based Evidence Distribution* – for distributing *trust certificates* in MANETs. The problem of trust evaluation is not addressed in the study. However, it contains a bio-inspired trust evidence distribution part which is responsible for distributing trust values to the entities of the network. Trust evidence comes in the form of certificates which are signed by their issuer's private keys. The content of the certificate depends on the trust metrics and the trust computation model. The authors assume that there are limited numbers of trusted certificate issuers, that their public keys are known and authenticated and that their private key cannot be compromised. As the number of signers is limited, the public key authentication can be done off-line before network's setup; therefore, the signers do not need to be online.

ABED is a reactive trust evidence distribution model. Ants are sent out only when a certificate is required. Every node has a certificate table in which every entry corresponds to one certificate and to the probability of selecting the next neighbor toward the destination. Two different kinds of *forward ants* are considered. *Broadcast ants* are sent out when there is no preference to the neighbors, for example, when there is no entry in the certificate table for the requested certificate. *Unicast ants* are sent out to the neighbor with the highest pheromone value.

Once a forward ant finds the required certificate, a *backward ant* is generated. Backward ants traverse exactly the way of forward ants towards the source node and update the pheromone value of each intermediate node. On the path of backward ants, the certificate is cached in every node on the path back to the source. Consequently, trust certificates are distributed and this assures the availability of certificates even if the signer is out of reach. After a period of time, since the probability of obtaining the certificates from neighbors rises, the certificate request overhead will be reduced.

ABED suffers from some disadvantages. One weakness is the assumption of existing, well known and authenticated signers. This assumption does not satisfy the dynamic nature of mobile ad hoc networks where nodes may join or leave the network. For a new node to join the network, continuous, secure availability of signers is needed in order to issue certificates and authorize the node's public key.

4.6 Proposed ACO Characteristic

In this thesis, we apply the ant colony optimization for trust evidence distribution. However, there are important differences between our work and the state of the art.

In [36] evidence is given in the form of a trust certificate that contains a public key, authorized by a signer’s private key. For example, through a certificate issued from A to B, A authorizes B to read or access data. The authors assume that there is a limited number of signers in the network, that their public keys are well known and authenticated and that their private key cannot be compromised. In our work, however, no such trusted signers exist; rather, every node in the network can play the role of a certificate issuer. As there are no predefined trusted certificate authorities, the system itself should be able to discriminate a trusted signer from malicious signers who aim at disrupting the authentication process.

In [90] the authors assume that each peer provides a unique peer identifier: this differs from our work in that we do not have such assumption.

Thus, due to the absence of such trusted signers, we need a mechanism to prevent malicious behavior and push the nodes to create a unique identifier (single public – private key) and to stay faithful to their identifier.

In this work, through applying mechanisms like general pheromone updating, punishing malicious nodes and rewarding honest nodes, the model tries to update the trust value of the nodes in such a way that nodes are persuaded to act properly. Through punishing mechanisms, misbehaving nodes will be isolated and will not be involved in authentication processes.

As a result, with respect to the existence of malicious nodes in the network, the model is able to retrieve the requested public key certificates through honest signers. The ACO algorithm applied in this thesis is inspired by the ACO algorithm used in TACS [3]. In there, a trust-based model was proposed that would find a most trustworthy service provider in peer-to-peer networks.

As we are, so we do;
and as we do, so is done to us;
we are the builders of our fortunes.
-Emerson

Chapter 5
Proposed Self-organized Authentication System Description

In non-centralized systems, using a path of trusted intermediate nodes is unavoidable for authentication processes. In this well-known technique for authenticating entities each node authenticates the next in the path. In order to have a more confident process, multi-path authentication techniques have been proposed [93]. But because of misbehaving nodes along the paths, ambiguities in correctness of certificates and, consequently, inaccuracy of public key-identity binding of the target node, the success of these approaches remains uncertain. Determining the owner of a public key is a basic requirement for executing secure communication in a system lacking a central authority. Therefore, systems like DSSA [94], SPX [95], PEM [96] and PGP [14] use the chain of authorities in the authentication process. In this model the source node can authenticate the first authority in the path; each entity in the path can authenticate the next authority in the path to the last entity, which is owner of the key of interest. The trust level of the source node to the obtained public key-identity binding relies on the correctness of each authority along the path. If even one node in a path incorrectly authenticates the next entity, the source can then be misled by the authentication of intermediate nodes, including the destination. In order to increase the assurance of the destination's authentication, it is proposed to use multi-paths.
This chapter we consider a mobile ad hoc environment in which all nodes perform five main processes to authenticate the destination node, for the purpose of securing communication. Different processes are applied to obtain a trustful multi-certificate chain. These processes are: public-private key generation and certificate issuing, certificate chains discovery, public key authentication by certificate verification, certificate chains trust updating and certificate revocation. In the following the details

of each process are illustrated. In non-centralized system, using a path of trusted intermediate nodes are unavoidable for authentication process. In this well-known technique for authenticating entities each node authenticate the next in the path. In order to have more confidant process, multipath authentication techniques have been proposed [93]. But because of the misbehaving nodes through the paths, ambiguities in correctness of certificates and consequently inaccuracy of public key-identity binding of the target node, the success of these approaches may be uncertain. Determining the owner of a public key is a basic requirement for executing secure communication in system with the lack of central authority. Therefore systems like DSSA [94], SPX [95], PEM [96], PGP [14] use the chain of authorities in the authentication process. In this model the source node can authenticate the first authority in the path; each entity in the path can authenticate the next authority in the path until the last entity which is owner of the key of interest. The trust level of the source node to the obtained public key-identity binding is relies on the correctness of every authority on the path. If only a node in a path incorrectly authenticates the next entity then the source can be misled in authentication of intermediate nodes including the destination. In order to increase the assurance of the authentication of the destination using multipath are proposed.

In this thesis we consider a mobile ad hoc environment, in which all nodes perform five main processes to authenticate the destination node in order to have a secure communication [53] [54]. Different processes are applied to obtained trustful multi-certificate chain. These processes are: public-private key generation and certificate issuing, certificate chains discovery, public key authentication by certificate verification, certificate chains trust updating and certificate revocation. The following shows the details of each process.

5.1 Public-Private Key Generation and Certificate Issuing

First each node creates its public key and corresponding private key locally. Then all neighboring nodes issue public key certificates for each other. If node A, based on its knowledge believes that a given public key PK_B belongs to a given node B, node A has to issue a certificate for node B and signs it with its private key Pr_A , to show its assurance of the binding of identity B, ID_B, to its related public key PK_B. Each node saves the public key certificates it issues for others in its repository. For every node in the certificate repository there is a confidence level of trust that shows to which extent that node issues correct and not mismatched certificates. Fig. 5.1 is an example of public key certificate generation for the nodes who are in each other’s radio range.

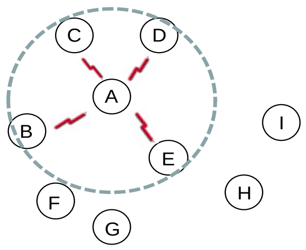


Fig. 5.1. Certificate issuing for neighboring nodes located in the radio range

Table 5.1 shows the certificate table (CT) in which every node stores the certificates issued for neighboring nodes. Each entry in CT corresponds to one public key and each column shows the belief of each neighboring node to a certain public key.

Table 5.1. Certificate table of node A

Certificates	Neighbors			
	B	C	D	E
A	$Cert_{A \to B}$	$Cert_{A \to C}$	$Cert_{A \to D}$	$Cer_{A \to E}$

Each node also has a table to store trust values of its neighboring nodes. Since this value presents the pheromone we name the table a trust-pheromone table (Table 5.2).

Table 5.2. Trust-pheromone table of node A

Trust-Pheromone	Neighbors			
	B	C	D	E
Pheromone	t_{AB}	t_{AC}	t_{AD}	t_{AE}

Having a sufficient connected certificate graph and also a fast creation of the certificate graph mainly depends on the motivation of the nodes for issuing certificates. In this work we assume that nodes in the network are motivated enough in order to create the certificates. They are assumed not to be selfish and are cooperative in certificate issuing task.

5.2 Initialization Phase of Certificate Issuing

We assume that during the network initialization phase, a direct trust relationship have been established between nodes and their physical neighbors. For fulfilling this phase, each node issues a public key certificate for each of its direct neighbor and signs the certificate with its private key. During this phase trust evidence are exchanged (e.g. neighbors' public key certificate). Consequently, based on the validity of the exchanged evidence, nodes assign a trust value to the certificate they issue. In this phase nodes do not have any information about the validity of the exchanged evidences. As a result, the initiated trust/pheromone value to all certificates are considered as a threshold value, t_{th} . In our trust model, trust value is defined as a continuous value in the rage of [0, 1]. The trust threshold is defined as

$$t_{th} = 5.5 \tag{5.1}$$

Any public key certificate's trust/pheromone value more than t_{th} represents a trustworthy node and any trust value less than t_{th} corresponds to an untrustworthy node. Assigning all certificates with threshold value at the initialization phase is caused by the complete uncertain situation at the beginning stage of the network establishment.

5.3 Certificate Chain Discovery

Our model is a reactive evidence distribution scheme. Ants are sent out only when a certain certificate is required. We assume that in the initialization phase, where during the key generation and certificate issuing step, trust relationships have been established between nodes and their neighbors. In our proposed scheme the certificate chain discovery process contains of two phases: forward phase and backward phase which will be explained in details.

5.3.1 Applied ACO

Mobile agents/artificial ants spread through the network from source to destination in order to find the most trustworthy certificate chains to obtain the public key a destination node. The ants remember the visited nodes they passed, and deposit 'pheromone' on them. Ants are attracted to paths with higher pheromone concentration. When an ant wants to move from a starting node S toward a destination it chooses one of the neighboring nodes of S, i, with the probability defined by following transition rule. This action is continued until finding target node.

$$p(\mathrm{S},\mathrm{i}) = \frac{[\tau_{\mathrm{Si}}]^{\alpha}.[\eta_{\mathrm{Si}}]^{\beta}}{\sum_{\mathrm{j}\in\mathrm{N(S)}}[\tau_{\mathrm{Sj}}]^{\alpha}\cdot[\eta_{\mathrm{Si}}]^{\beta}} \quad ; \quad \sum_{\mathrm{i}\in\mathrm{N(S)}} p(\mathrm{S},\mathrm{i}) = 1 \tag{5.2}$$

where τ_{Si} is the pheromone deposit on the edge between S and i, η_{Si} is the goodness value of the link between S and its neighbor node, $\mathrm{N}(S)$ is the list of

neighboring nodes of S and α and β are the weights for balancing the deposited pheromone and the goodness value of the edge respectively.
The following transition rule is used to provide a pseudo-aleatory path choice:

$$r = \begin{cases} \operatorname{argmax}\ j\epsilon N(S)\left[\tau_{Sj}\right]^{\alpha} . [\eta_{Si}]^{\beta} & \text{if} \quad q \le q_0 \\ R & \text{otherwise} \end{cases} \tag{5.3}$$

where r is the next chosen node by an ant in its next movement, q_0 is the probability of choosing deterministically the most promising edge, q is a measure in range of [0, 1] and R is a randomly selected neighbor node.

5.3.2 Forward Phase

In forward phase or certificate request phase, source node S sends out several ants to explore the path to obtain the public key of the destination node, D (Fig. 5.2). S sends forward ants in form of a certificate request packet to nodes that it directly trusts as a message like: $S \to B : \{cert\_req\}$.

An ant at each intermediate node, e.g. *B*, chooses, the next node e.g. *F*, according to formula 5.2, in which τ_{BF} is the pheromone which represents the referral trust that node *B* has on relaying node *F*. We assume that with the probability of q_0, neighbors with pheromone value more than the trust threshold should be chosen. η_{ij} is the functional trust that node *i* has in general toward node *j* in providing the authentication service. In this work we assume that all nodes are trustable in making the authentication service available. The process is continued until the forward ant arrives at destination.

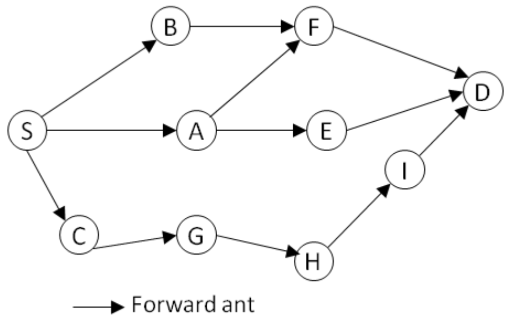


Fig. 5.2 Forward ants carry certificate request

5.3.3 Backward Phase

Backward phase or certificate reply phase in which the destination sends backward ants towards the source node. After receiving a forward ant at the destination D (Fig.

5.3), it generates a backward ant and sends it to its previous node in the forward path, e.g. node F in (Fig. 5.3). This backward ant is the certificate reply message: $D \rightarrow F : \{cert\_rep\}$.

A backward ant retraces exactly the path of the forward ant back to the source. Through returning of the backward ant from the destination to the source, each intermediate node adds certificates into reply packet and provides a chain of certificates (Fig. 5.3). For example, node F searches for the certificate of the destination, $Cert_{F \rightarrow D}$, in its certificate repository, adds it to the reply packet and sends it to B: $F \rightarrow B : \{cert\_rep, Cert_{F \rightarrow D}\}$.

Node B receives this reply packet from F, searches for the certificate of F, adds it to backward ant and sends it to S as following message:

$$B \rightarrow S : \{cert\_rep, Cert_{F \rightarrow D}, Cert_{B \rightarrow F}\}$$

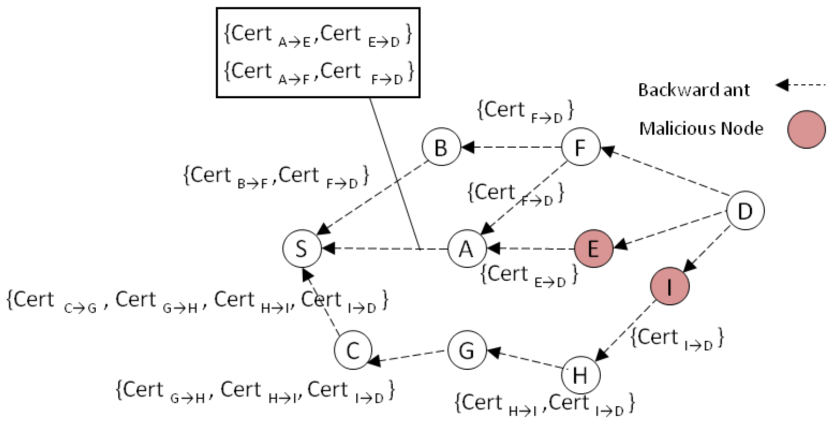


Fig. 5.3. Backward ants carry certificate reply

Finally the source node receives the backward ants from the destination, computes the trust value of the chains, using formula (3.1) and recovers the public key of the destination. The source node inserts the certificate chains and their corresponding trust values in its certificate chain table (Table 5.3).

Table 5.3. Certificate chain table of a node

Certificate chain	Destination	Trust/Pheromone value
SBFD	D	t_1
SAED	D	t_2
SAFD	D	t_3
SCGHID	D	t_4

Each intermediate node on the path keeps the part of certificate chain discovered from its own to the destination in its certificate chain table. This allows the nodes for further analysis to cross-check to reward the honest certificate chains and punish the chain contains adversaries. Certificate chains with malicious nodes will be kept for further analysis.

5.3.4 Certificate Chain Discovery Latency

The certificate chain discovery latency of the proposed bio-inspired authentication scheme consists of the latency of two forward and backward phases in certificate chain discovery process.
As in the forwarding or requesting phase ants traverse several paths through most trustable relaying nodes, the *maximum* latency of this phase in form of delay of certificate discovery, ($\mathrm{L_{forward}}$), can be calculated as follows:

$$\mathrm{L_{forward}} = \sum_{a=1}^{N_{ant}} \left(\sum_{i=1}^{hops} \mathrm{T(Size_{cert\_req})} \right) \tag{5.4}$$

where N_{ant} represents the whole number of ants sent out by towards destination, $hops$ is the number of hops which an ant passed to find the target node. $T(Size_{cert\_req})$ is the time needed to transferred certificate request. This time includes: delays caused by buffering during chain discovery; delay of waiting, for a predefined limited of time, for a trustworthy neighbor; retransmission delays while timeout happen; MAC layer delay caused by channel status checking and waiting for channel. During the forward phase only the certificate requests are transmitted.

In backward phase, the destination node does not add its certificate into reply packet; because the destination's certificate was stored in the direct previous neighbor from which the forward ant come from. The *maximum* latency is calculated as follows:

$$L_{backward} = \sum_{a=1}^{N_{ant}} \left(\sum_{i=1}^{hops} T(size_{cert\_rep} + (i-1).size_{cert}) \right) \tag{5.5}$$

Where $Size_{cert-rep}$is the size of the certificate reply and $Size_{cert}$ is the size of each certificate.

The total latency of the certificate chain discovery phase is calculated as follows:

$$L_{total} = L_{forwrd} + L_{backward} \tag{5.6}$$

5.4 Public Key Authentication by Certificate Verification

After reception of several certificate chains the source node investigates the *validity* and *correctness/authenticity* of each certificate chain by verifying each certificate using the public key of its immediate next hop in the chain. Validity checks if the certificates in the certificate chains are not expired. Correctness, investigates that the certificate contains the correct or fake user-key binding. Verification of the certificates in a chain will be performed using the public key of the previous certificate in the chain. For example we assume S receives SBFD as certificate chain (Table 5.2). Then it uses the public key of the node B, PK_B , to verify the first certificate in the certificate chain, $Cert_{B\rightarrow F}$. When verification of $Cert_{B\rightarrow F}$ is successful, the source node retrieves the public key of F, PK_F , through certificate $Cert_{B\rightarrow F}$ and then verifies the next certificate in the chain, $Cert_{F\rightarrow D}$. In case of having any discrepancy during verification phase, the certificate chain will be discarded and all nodes along the certificate chain will be punished.

If S detects no inconsistent certificates, it rewards all the nodes along the certificate chain and regards the maximum received trust value as trust value of the D's certificate:

$$t_{SD} = max_{i \in CCH(S \rightarrow D)}(R_i) \tag{5.7}$$

where $CCh(S \rightarrow D)$ is the list of certificate chains which S receives after request for the public key of D. R_i is the reliability of the honest chan. The reliability of the chain depends on the number of ants that traverse that path and the trust value of every intermediate node. It is calculated as follows:

$$R_i = ant_K\,\% \,.\; t_{SD} \tag{5.8}$$

Where ant_K % is the percentage of ants that traverse along this chain and t_{SD} is the trust value of the chain calculated as formula (3.1).

5.5 Certificate Chains Trust Update

Trust updates will occur in the four following situations:

a. General local updating: In each intermediate node, if there is no mismatch information received by backward ants from its neighbors, the pheromone entry of the neighbor node, from where backward ant came from, will be updated as following:

$$t_{ij} = (1-e).t_{ij} + dt_{ij} \tag{5.9}$$

$$dt_{ij} = e.\left(1 + d_{jD}\ .(1-e).(1 - t_{ij}.\eta_{ij})\right).t_{ij} \tag{5.10}$$

where e is the pheromone evaporation value. By dt_{ij} in formula (5.10) we give the opportunity to edges with lower values of pheromone to recover faster. d_{jD} is the distance, hop count, between j and the destination. In our work we assume that every node, in the certificate chains, updates only the trust value of its direct neighbors from which backward ant has come. We borrowed the general local updating equation from TACS [3].

b. Punishing: In case of observing any mismatch in certificate chains, node *S* performs the punishment. It reduces the trust level of its neighbors who located on a certificate chain led to the fake certificate. Punishment process is performed by evaporating the pheromone of the edge between *S* and those neighbors (e.g. *A* and *C* in Fig. 5.3). Node *S* creates a *punishing-update message* signed by its private key and sends it to the neighbor in the mismatched certificate chain; and punishes these neighbors as follows

$$t_{ij} = t_{ij} - \omega.e.t_{ij}.dp \quad , \quad dp = \frac{1}{D_{PD}} \tag{5.11}$$

Weight D_{PD} is the distance factor (hop count) between punished (P) node and destination node (D), which can be obtained through the certificate chain. The longer this distance the less is the punishing amount. ω is the trust value of a node toward its notifier neighboring node (e.g. t_{AS} in figure 5.3). This weight is equal to 1 if the notifier itself is also the punisher. Locating in node *R*, in order to calculate the ω , all received *punishing-update-messages* corresponding to a unique Public key request should be considered. For a request for an aimed *PK_i*, *R* waits to receive all *punishing-update-messages*, related to PK_i , from its neighbors. Subsequently *R* calculates the ω_{RK} as follows:

$$\omega_{RK} = \sum_{K\epsilon N_R} {}^{t_{RK}}\!/_{|punishing-update-messages|} \tag{5.12}$$

where *K* is all neighbors of *R* who, in the chain discovery process of PK_i , send the *punishing-update-messages* to *R*. When a node receives a *punishing-update-message* it verifies the validity of the message decrypting the message with the public key of the issuer of the message. Furthermore node checks for the existing of the certificate chain in its certificate table that the received *punishing-update-message* is correspond to. *Punishing-update-message* referred to no-existing certificate chain will be ignored.

The punished node with trust value less than trust threshold will be isolated and not used further in the authentication process.

All nodes in the chain (e.g. A and C) will also punish their next immediate neighbor in the path towards the destination.

c. Rewarding: Every node will reward its next hop along the reliable certificate chain and update its trust value as follows:

$$t_{ij} = (1-e).t_{ij} + e.\left(1 + R_i.\eta_{ij}\right).t_{ij} \tag{5.13}$$

where $reliability_i$ is the reliability of the certificate chain. It shows the edges with higher pheromone value are more rewarded than those with lower value. A node receives a *rewarding-update-message* encrypted by issuer of the message in order to check the validity of the message.

d. Mobility update: Node mobility in the network results in establishing new trust relationships. The trust value of the new neighbors will be initialized with the trust threshold value. Trust value of old neighbors will be evaporated during the time:

$$t_{ij} = (1-e).t_{ij} \tag{5.14}$$

Trust value of Trustworthy old neighbors ($t_{ij} > t_{TH}$) will be decreased at least to trust threshold value. Mobility enables nodes to add more certificates in their certificate tables and consequently facilitates certificate chain discovery phase.

Trust values defined a measure between 0 and 1. After each updating, trust value will be normalized. Minimum trust value in the system considered as $t_{min} = 0.0001$ and maximum trust value is $t_{max} = 0.9999$. After trust update phase, if a trust value exceeds these limits it will be set to the corresponding boundary value.

5.6 Certificate Revocation

If an issuer node believes that in a certificate it issued, the binding between the ID and public key of the target node is no longer valid or the trust value of it is less than the trust threshold, t_{th} , it can revoke that certificate. Moreover it signs the certificate revocation by its private key and sends it to its neighbors. When a node receives a certificate revocation, first it verifies the signature of the revocation message. If it has the trust value of both the sender of the revocation and that of the node its certificate is claimed to be revoked, it compares these trust values. If the sender is untrustworthy and its trust value is lower than t_{th} then the receiver ignores the revocation message. Otherwise it calculates the following value:

$$t_{it} = t_{it} - t_{is} \tag{5.15}$$

where t_{it} is the trust value of the node toward target node that its certificate claimed to be revoked, t_{is} is the trust value of the node toward sender of the revocation message and t_{st} is the trust level of target node claimed by sender of revocation message. If t_{it} is lower than trust threshold, it deletes the revoked certificate, otherwise the certificate is still to be used.

5.7 Security Analysis

In this chapters we proposed the autonomous authentication using a chain of trusted intermediate nodes, each able to authenticate the next in the chain. In self-organized public key management, the signature of a public key certificate will be verified by the public key of the certificate issuer. Because there is no central global authority to confirm the validity of the public keys there should be policy to control and maintain the validity of the public keys. In a certificate chain a public key certificate would include a digital signature to make the recipients assure about the validity of the public key. However in this model there can be attackers who try to defect the certificate chain discovery and consequently the authentication process. In chapter 3 different kind of attacks, that threat the security of the certificate chain model, were explained. In general these malicious behaviors can be considered as followings:

- Attackers who insert false certificate in the certificate chain
- Attackers who deceive the other nodes and make them to believe in false certificates.

In this thesis we assume a malicious node as a node who inserts false certificates in the certificate chain during the discovery phase.

In Fig.5 if the malicious node G issues a false certificate for H and bind an incorrect public key to ID_H, therefore the certificate chain verification process in source node will be failed. In our proposed model the reliability of certificate chains passing through malicious node G fading during the time. In results this node will not be chosen more for certificate chain discovery phase.

However authenticating through a single certificate chain is vulnerable to the duplicating. If source node S, through a certificate chain receives a certificate that binds PK_{D1} to node D, and then later he receives another certificate chains through that node D is bounded to PK_{D2}, then duplication is happened. In this case all mismatched certificate chains are discarded and no one is used. In other case if a node, e.g. node A in Fig. 5.4, is a malicious node who impersonates the destination node, therefore every certificate chain contains A, is not trustworthy. This problem can be solved through *trust independency*.

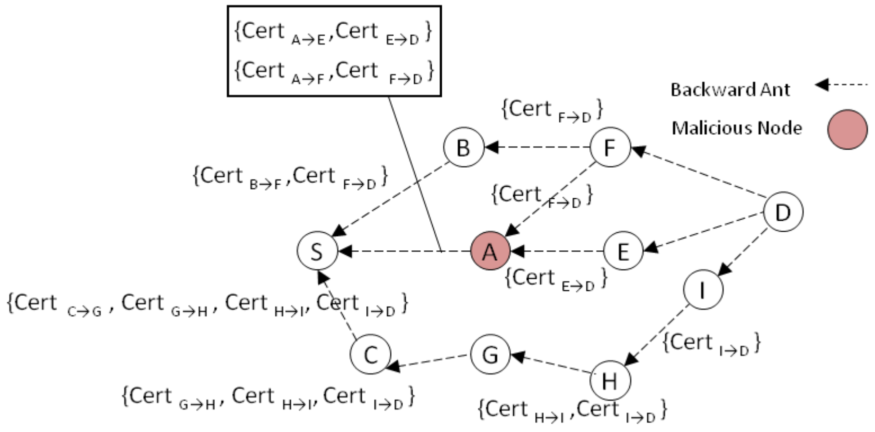


Fig. 5.4. Every intersected certificate chains with malicious node here A, is untrustworthy

These problems can be resolved through finding two independent certificate chains. In the authentication process, discovery of independent certificate chain results in an increase the reliability of the public key of a target node. In [97] the authors propose approximation algorithm for find set of independent paths support authentication in PGP. A self-organized public key management for mobile ad-hoc networks is presented in [18] using at least two independent certificate chain for each source and destination pair.

For providing trust independency, first the received public key certificate chains will be set to different clusters, each cluster report the same public key for the destination. Then in each set disjoint certificate chains will be found and the public key returned by majority of independent path will be accepted. Although independent certificate chain can increase the trustworthiness of the reported public key, but there still exist drawbacks against *Sybil attack*.

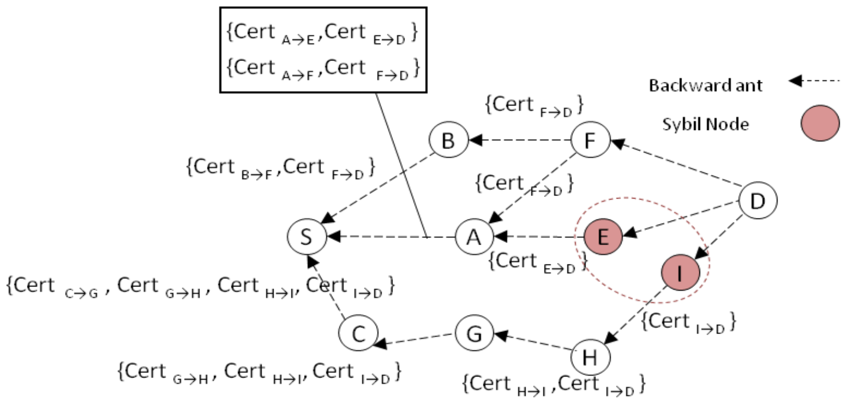


Fig. 5.5. Node E is a Sybil node with a fake identity I.

Fig. 5.5 demonstrates that E is a *Sybil* node with other self-created identity I. In order to report the public key of destination D, they bind a false public key PK_{sybil} to ID_D. In certificate chain discovery phase although the source node S receives two independent paths however they don't transfer the valid D's public key. Cooping this attack requires knowledge discovery in received certificate chains. *MANET Mining* [98], which is not in the scope of this work, enables the extraction of implicit, previously unknown information and hidden relationships among nodes. The aim of this technique is to seek and to identify which misleading nodes are related to each other who try to make other nodes believe in false certificate. As *Sybil* attack is a serious threat for a self-organized authentication mechanism, in our future work we aim at detecting Sybil attack. In this regards we plan to aggravate our proposed ant-based authentication mechanism with a certificate chains mining mechanism in order to find the hidden relationships between nodes through different certificate chains. Therefore a source node, by analyzing the certificate chains that ants bring back to it in discovery phase, can be able to verify the *Sybil* nodes laying trough certificate chains.

5.8 Summary

In our proposed self-organized authentication mechanism every node creates its own public-private key pairs and lets its public key be known to its neighbors. Nodes issue public key certificates for their neighbors. In the bootstrapping phase, when there is no trust relationship between nodes, the initial trust value (0.5 in our work) is assigned to each of the issued certificates. Nodes issue certificates only for their neighboring nodes. The construction of trust opinions and trust dissemination will be performed during the reactive certificate chain discovery phase. In the discovery phase, forward ants try to find paths to the destination according to the density of pheromone/trust values between neighbor nodes. The source node, after receiving multi certificate chains, verifies them to exclude the chains with false certificates. We consider the malicious behavior as inserting a false certificate in the chain. The trust value of nodes of honest chains will be reinforced, while nodes of chains with false certificates will be punished. Therefore, with the passing of time, the malicious node will be isolated from being chosen in certificate chains. In our work, once a certificate is issued, it will stay in the network. We assume that a certificate expiration time is longer than the simulation time. By revocation, when the trust value of a public key certificate goes below the threshold value, it is assumed to be a revoked certificate and will not be used anymore in the certificate chain discovery phase.

Chance does not exist.
There is always a cause and a reason for everything.
-Ostad Elahi

Chapter 6
Simulation and Results

In order to investigate the efficiency and robustness of our self-organized authentication mechanism in an ad hoc environment we simulated the scheme. In this regard, we explore the behavior of our design under nodes' mobility and relating to the existence of malicious nodes in the network. For this purpose we consider malicious behavior as a node which inserts a fake certificate into a certificate chain during the certificate chain discovery process. A malicious node issues a certificate and binds a false key to an ID. This kind of misbehaving will be realized by the source node in the certificate chain verification process. During the certificate chain verification, if the source node reaches a mismatch of key and signature in the chain, it will ignore the verification of the rest of the certificate chain and will punish its direct neighbor which led to this certificate chain. A punishment message is spread to the certificate chain via each intermediate node. We assume that each intermediate node keeps the backward ants' traces from destination to source. It enables intermediate nodes to identify their direct neighbor through the malicious certificate chain and to punish them by updating the pheromone value of their direct neighbor's on the chain. The simulation of the proposed ant-based self-organized authentication mechanism is performed in two parts.

6.1 Simulation Environment

First, we used Matlab for the simulation and evaluation of our design. We use the MAC layer protocols 802.11 with data rate of 5 Mbps. 30 Nodes were randomly placed in a 1000x1000 m^2 square area. Communication is in one hop distance with a radio range of 250 meters. Random waypoint mobility was considered as a mobility pattern with maximum speed of 0m/s (static network), 2m/s and 10m/s with a pause time of 3 seconds. At the beginning of the simulation, the nodes' uncertainty about their neighbors' trustworthiness was at the maximum level. Therefore, in the initialization phase, the trust values between a node and its neighbors were taken to be equal to the trust threshold value (equal to 0.5). Due to this initial value, all neighbors of a node can be potentially chosen in initial requests, since the node's environment is unknown.

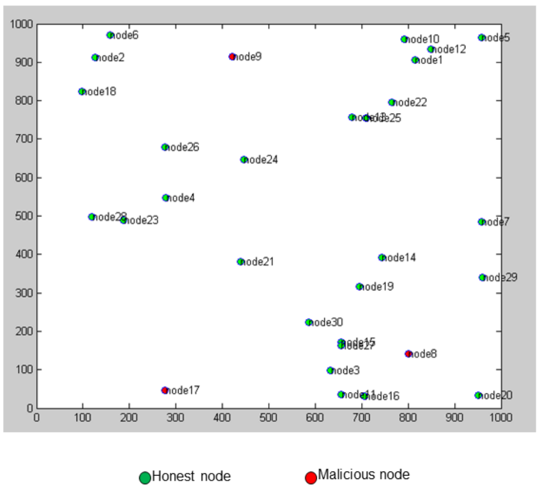


Fig. 6.1 Example of a scenario consisting of 10% malicious nodes

Table 6.1. Simulation parameters

Parameter	**Values**
Number of nodes	30
Environment	1000x1000m^2
Radio range	250m
Mobility model	Random waypoint
Speed	{0,2,10} m/s
MAC layer	802.11
Number of ants	{5, 10,15, 20, 25, 30, 35}
Trust threshold	0.5
e	0.1
q_0	0.90
$\alpha = \beta$	1
η_{ij} $for \forall i, j$	1

During the simulation and application of pheromone updating, the trust values of malicious nodes converge to a value lower than the threshold, while the trust values of honest nodes approach a value higher than the threshold. Consequently, the pheromone updating phase reduces uncertainty so that neighbors with higher trust values will be chosen during the certificate chain discovery phase.

Each run proceeded within a maximum of 600 seconds and in several iterations. We assume that five requests for public keys of random honest destinations are generated in each iteration. Requests were generated sequentially. After processing each request and obtaining the public key of the destination, 5000 packets of data were transferred and the pheromone values were updated. In our simulation, a public key certificate was considered to be 512 bytes in length, according to the concept that encrypted data with a key size of 2048-bits should be secure. For the result in each iteration we considered the average of all five requests. After each round the updated pheromone values were launched to the network and new random requests were created. Fig.6.1. shows a scenario entailing 10 percent malicious nodes. In table 6.1 the simulation parameters are summarized.

6.1.1 Performance Evaluation Metrics

In order to evaluate the performance of the proposed scheme we have selected the following three metrics:

- *Success rate of certificate chains discovery*: the percentage of requests for which the requester successfully obtains the public key certificate. This is the number of requests which obtained correct certificates over the total number of requests.
- *Ratio of reliable certificate chains*: this is the number of received certificate chains that succeeded in the verification phase via the source node, over the number of transmitted ants that have found the destination.
- *Reliability of certificate chain*: the average of reliability of certificate chains for each request.
- *Certificate chain discovery latency:* we consider the average end-to-end delay during certificate chain discovery as the certificate chain discovery latency metric. It includes all delays at MAC and physical layer.

During the chain discovery phase, a backward ant waits for the duration of one second to account for possible topology changes. When a forward ant arrives at the destination and completes its certificate chain, it will try to follow the same path backwards to the source node. After the waiting period the backward ant either takes the hop to the same node it came from, in case the node is still in communication distance, or it is considered lost. For each forward ant, the source node awaits a corresponding backward ant for a timeout period. We set this time as 0.2 seconds in our simulation. The source node has a re-transmission limit of three forward ants, when a timeout occurs. In case of a timeout, which may be caused by topology change, the destination node also sends not more than three backward ants.

6.1.2 Investigating the Effect of the Number of Ants

In the first part of the performance evaluation we investigated, on the predefined metrics, the effect of the number of ants sent out from source to destination for each authentication request. Simulation runs have been performed for different numbers of ants: 5, 10, 15, 20, 25, 30 and 35. The simulation results, for each number of ants, are averaged over all iterations in one run and are also averaged over several runs. Simulation was repeated for three runs in this part. We assumed 10 percent of the whole network to consist of malicious nodes.

Fig. 6.2 shows that, increasing the number of ants from 5 to 20, the success rate of certificate chains increased. We consider the fluctuation in the diagram for mobile network with speed 2m/s and 10m/s, at the point with ant number 15 and 20 as because of the random nature of the scenarios. However, this rate decreased after applying more than 20 ants in the static network and mobile network with 2m/s; it stayed unchanged in the dynamic network with speed of 10m/s. The results therefore show that a continuous increase of ants does not increase the success rate of certificate chains significantly, in both static and dynamic networks.

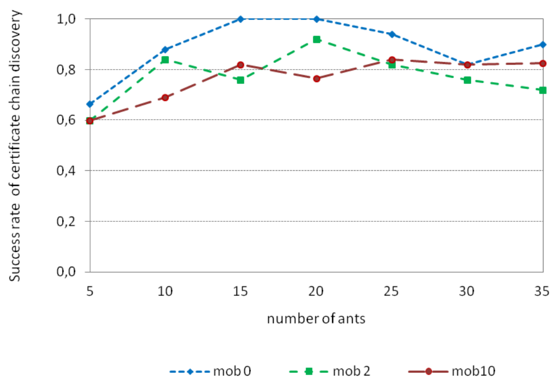
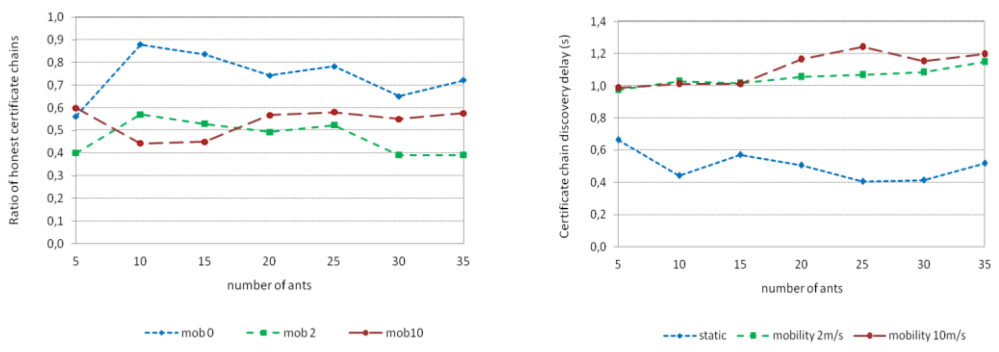


Fig. 6.2. Success rate of certificate chain discovery by varying number of ants

Fig. 6.2 also states that the success rate of certificate chain discovery is reduced for mobile networks compared to static networks. We reason that more backward ants get lost due to topology changes.

Fig. 6.3 a) shows the ratio of the reliable certificate chain with varying numbers of ants. In both static and dynamic networks, increasing the numbers of ants for each authentication request does not increase the ratio of reliable certificate chains.



(a) (b)

Fig. 6.3. a) Ratio of reliable certificate chains by different number of ants. b) Average certificate chain latency of an authentication request against different number of ants

Further, we investigated the effect of the number of ants on the average certificate chain discovery latency of the authentication requests (Fig. 6.3(b)). It is considered to be the needed average time for a certificate chain discovery phase. This value is the average of end-to-end delay of control and data packets necessary for a public key authentication request. It demonstrates the end-to-end delay of spreading forward- and backward ants and the required certificates which are conveyed by backward ants from the destination to the source node. It considers all dropped or successful packets reaching the destination. The results show that the average discovery latency for all requests, in a static network, decline with the number of ants. More ants can find the requested public key faster. However, end-to-end delay in dynamic networks is not reduced, because of topology changes. Topology changes cause loss of more ants during the certificate chain discovery phase.

Fig. 6.4 shows what impact the number of ants has on all performance evaluation metrics, considering three different mobility statuses, 0, 2 and 10 m/s speeds.
Evaluation metrics are: *success rate of certificate chains discovery,* s_{CH}; *ratio of reliable certificate chains,* rr_{CH}; and *reliability of the certificate chain,* r_{CH}; certificate chain discovery latency, l_{CH}. In order to maximize s_{CH} , rr_{CH} , r_{CH} and minimize l_{CH} we define a *global* performance evaluation metric as following:

$$global\ performance\ evaluation\ metric\ = (s_{CH}.rr_{CH}.\ r_{CH})\ /l_{CH} \qquad (6.1)$$

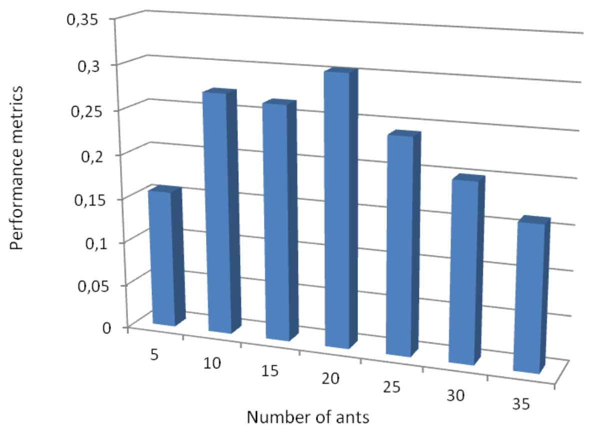


Fig. 6.4. Comparison of the effect of the number of ants on the evaluation metrics

The investigation of the effect of number of ants shows that increasing the number of ants is not a necessity for achieving better results. Considering our scenario, the experimental results show that sending out 20 ants achieves the better results when initiating an authentication request. For the remaining simulations we therefore used 20 ants for each request, since it reasonably corresponded to the performance metrics.

6.1.3 Self-Learning Process

Nodes in our self-organized authentication mechanism have no knowledge about their neighbors' behavior in the *bootstrapping phase*. Therefore, an evolutionary system is needed to gather information about the behavior of other nodes – the public key signers in the network. In our ant-based public key authentication model, we present this process of self-learning during the period of simulation time.

For this step we investigated the value of different metrics for each time interval. Like in the previous scenario we generated 5 random requests in each iteration (time intervals). The duration of each time interval was set to 100 seconds. We applied 20 ants for each request. Of all nodes in the network 10% were malicious nodes, and all nodes are static. Clearly, in the first iterations, since no knowledge about the trustworthiness and untrustworthiness of the neighboring nodes existed, the probability of choosing a malicious node along the certificate chains was higher. Therefore, the success rate of correct certificate chains discovery could be lower than in the higher iterations, where the trust values were updated. Fig. 6.5 shows that in our scenario only in the first time interval some public key certificate requests did not succeed.

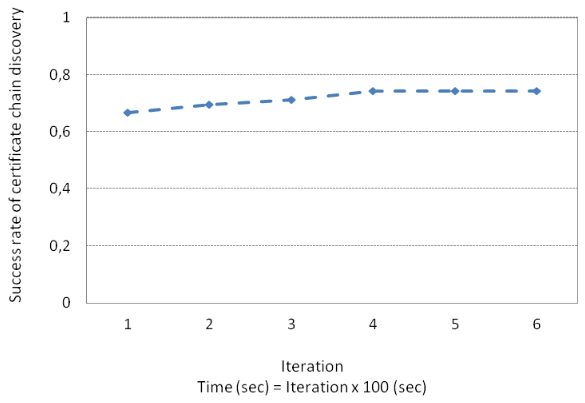


Fig. 6.5. The success rate of certificate chain discovery during simulation time

Starting from the second iteration, the success rate of certificate chains discovery is always one; however, Fig. 6.6 shows that the reliability of certificate chains is not the same in all iterations; rather, this value increases with time. By updating pheromone values, as time passes, nodes with higher pheromone values are chosen as public key signers through a certificate chain. By passing through time, step by step, the network learns how to choose more trustworthy, intermediate nodes as vouchers for the public key of a target node. During the simulation time, via updating pheromone values, ants were attracted to a chain with higher pheromone/trust value. This causes a mutual increase of pheromone values. Reinforcement of the pheromone value of trustworthy nodes increases the reliability of certificate chains in duration of time.

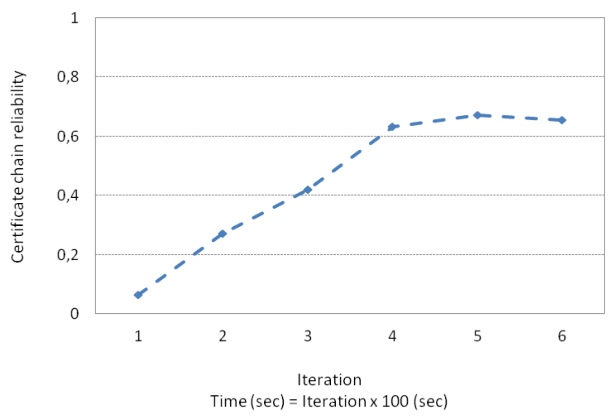


Fig. 6.6. Evolution of public key certificate chain reliability during simulation time

By incentive mechanisms, the pheromone value of trustworthy nodes has been increased. Consequently, in higher iterations most of the ants converged to the certificate chain with the higher trust value. As a consequence, the time of exploring honest certificate chains is reduced.

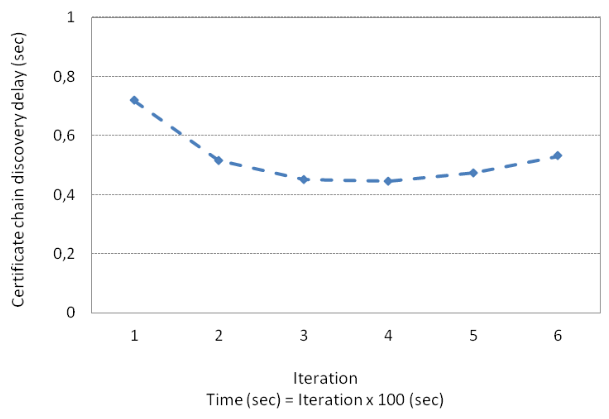


Fig. 6.7. Certificate chain discovery latency during simulation time

Fig. 6.7. shows the reduction of certificate chain discovery latency during the simulation time.

The results illustrate that the proposed ant-based public key management mechanism autonomously, and in a distributed manner, learn the environment. It chooses the trustworthy nodes in authentication processes. The model thwarts the effect of the malicious nodes, which attempt to insert fake certificates into the certificate chain, by avoiding selecting them in the certificate chain discovery phase. In results, the reliability of the discovered certificate chains is increased during the time. Learning process reduces the uncertainty about the trustworthy and untrustworthiness of the nodes in the network. Consequently in higher iterations honest certificate chains can be discovered in a shorter time.

6.1.4 Effect of the Malicious Nodes and Mobility

In this part the robustness of the proposed scheme against an increase of malicious nodes is investigated. Furthermore the effect of mobility is also explored. This investigation was performed in two parts. Matlab is the simulation environment for the first part. In second part we investigated the effect of the malicious nodes and mobility on the proposed model, with Qualnet network simulator environment.

6.1.4.1 Part I

For this section five different initial topologies were considered. Each topology was repeated for three different simulation runs. The results were averaged over all 15 runs. To explore the effect of mobility on our proposed scheme we run simulations for mobility with a speed of 2m/s and for 10m/s. In all scenarios the source node sent out 20 ants for each public key request.

We increased the number of malicious nodes from 0% to 100% of all nodes. Fig. 6.8 shows that the success rate of certificate chains in case of near 20% malicious nodes is above 0.5.

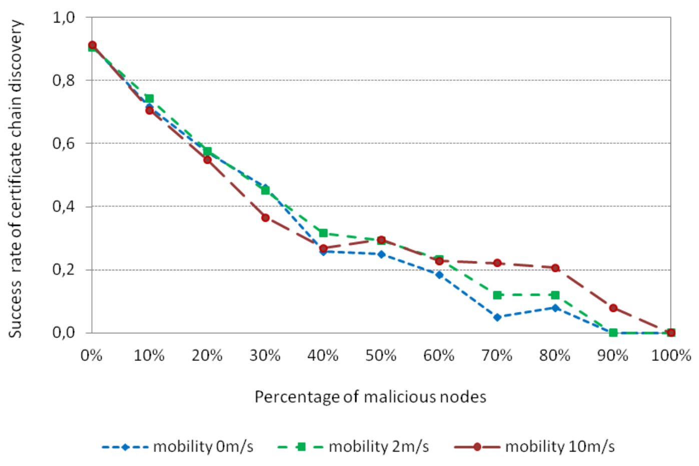


Fig. 6.8. Success rate of certificate chain

The results demonstrate that in networks with a lower number of malicious nodes, higher mobility leads to a reduction in the success rate of certificate chains. This could result from the fact that more backward ants get lost in networks with mobile nodes. However, in case of a higher number of malicious nodes in the network, here 40% of the whole network size, finding honest certificate chains is more successful. Although mobility causes, on the one hand, more breakage of the path of backward ants, on the other hand, mobility enables nodes to encounter new neighbors and increases the probability of finding trustworthy certificate chains. Therefore, mobility helps the survival of the network despite having a high number of malicious nodes.

Fig. 6.9 demonstrates the ratio of reliable certificate chains which is reduced when increasing the number of malicious nodes in the network. It shows acceptable results comparing a static network with a network of mobile nodes. Although mobility causes dropping backward ants by motion, a node can encounter new neighbors and reduce its uncertainty about the environment by updating its pheromone table. The ratio of honest certificate chains is considered as the fraction of the ants which find trustworthy certificate chains in the discovery phase. By having a higher number of malicious nodes in the network, here more than 60%, this value is higher in networks with mobile nodes compared to static networks.

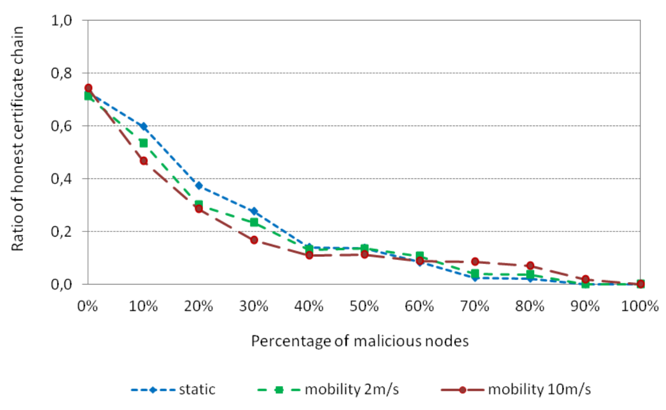


Fig. 6.9. Ratio of reliable certificate chains in case of increasing percentage of malicious nodes in the network

In the chain discovery phase nodes are chosen with a trust value higher than the threshold value; therefore, when the number of malicious signer nodes is increased, the possibility of finding correct certificate chains is decreased and leads to having less end-to-end delay. Fig. 6.10 shows the decrease in certificate chain discovery latency when the number of malicious nodes increases. This latency in networks with mobile nodes is higher than in static networks. The reason is the loss of ants due to topology change or due to waiting time for finding a neighbor.

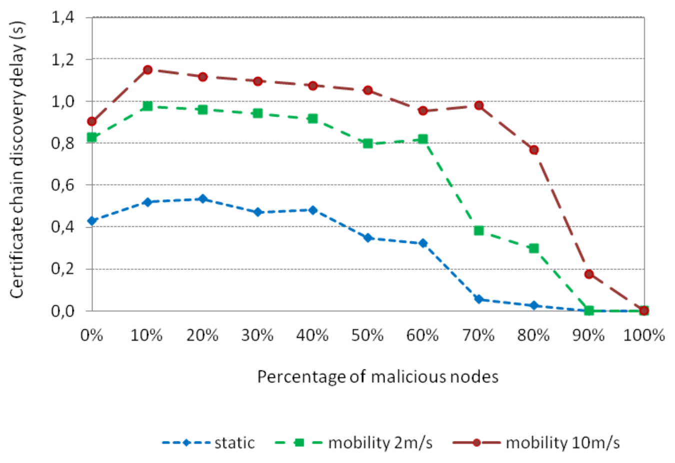


Fig. 6.10. Certificate chain discovery latency for different percentage of malicious nodes

In Fig. 6.8 the success rate of certificate chains in a network consisting of mobile nodes with speed 10m/s is about 0.2, despite having 70-80% malicious nodes in the network. This leads to difference certificate chain latency between 70-80% and 90% (Fig. 6.10).

6.1.4.2 Part II

In this part we aim at investigating the performance of the proposed model with the network simulator environment Qualnet. We use Qualnet to explore the robustness of our autonomous authentication mechanism in case of an increase in the number of malicious nodes in the network. Furthermore, the effect of mobility is investigated. Network area was set to 1500x1500m^2 with no obstacles that could influence the mobility or the signal propagation of a node. We considered 30 nodes with random placement in the area. The transmission range of each node was 250 meter. In order to find the neighboring nodes, the interval of broadcasting hello messages was set to 1 s; the hello message loss was set to two in order to find the disappeared neighbors, via movement..

For the transport layer, a UDP protocol with default parameters was used. For the MAC layer, we used the IEEE 802.11 protocol and data transmission rate of 2Mbps Data traffic was generated by a constant bit rate (CBR).

The complete simulation time was 600s in total. During this time 30 data sessions were run between randomly chosen source-and-destination pairs. For every run each node participated only once as a source node and once as a destination node. Each CBR session was started directly at the beginning of the simulation, and be continued till the end of the simulation. The data sending interval was set to 1s. Thus, one source node could generate, during one run, maximum 600 data packets with the size of 512 bytes. This is the worst case scenario since all nodes in the network are sending and

receiving packets during the whole simulation time. For each data transfer 5 ants were sent out to find the public key of the destination.
The size of the public key certificate is the same as the size of a data packet with a total size of 512 bytes in length. The buffer size in was considered for 100 packets. Table 6.2 lists the simulation parameters.

The random way point mobility model was chosen for mobility, with three different settings: 2m/s, 10m/s and 20m/s. The basic idea of the mobility model is that each node moves with the defined speed in a straight line to a random destination and then waits there for a pause time of 3s; then it moves towards the next random destination. We considered 10 different topologies. For each topology 5 runs were performed with different random CBR applications for 30 nodes. Performance metrics, calculated for each node, were averaged over all 30 nodes for each run. Finally, the results were averaged over all runs.

Table 6.2 Simulation parameters (Part II)

Parameter	**Values**
Number of nodes	30
Environment	1500 x 1500 m^2
Radio range	350 m
Mobility model	Random waypoint
Speed	{2, 10, 20} m/s
PK certificate size	512 bytes
Number of ants	5
Trust threshold	0.5
e	0.1
q_0	0.90
$\alpha = \beta$	1
η_{ij} $for \forall i,j$	1

6.1.4.2.1 Performance Metrics (Part II)

In order to evaluate the performance of our self-organized public key management approach we have selected the following metrics:

Number of certificates issued for neighboring nodes: this is the total number of certificates that a node has issued for its all encountered neighbors.

Average certificate chain length: this is the average hop length of all honest certificate chains selected by a source node during the total simulation time.

Control packets: this is the total number of bytes of control packets involved in the discovery phase, including forward ants, backward ants, trust update packets and select route packets.

6.1.4.2.2 Evaluation of the Simulation Results (Part II)

In this part we performed further investigations about the effect of mobility and the number of malicious nodes in our proposed authentication model.

Fig. 6.11 illustrates that the number of issued certificates by a node with higher mobility is higher than the number of initiated certificates by a node in static network or a node with lower speed. Nodes with higher speed have a greater chance to meet new neighboring nodes than those with slower speed. It can be observed that nodes with higher mobility can meet almost all nodes in the network during the simulation run.

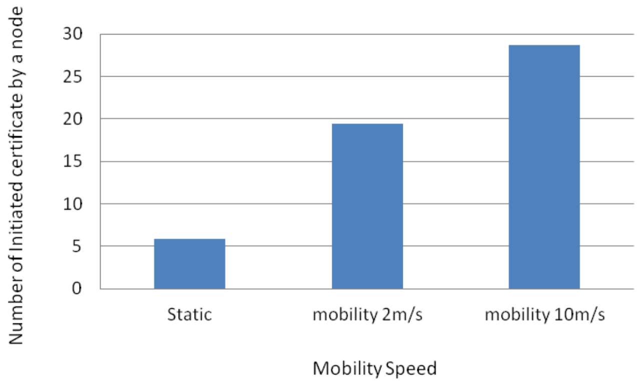


Fig. 6.11 Numbers of issued certificates by a node during the simulation time

The number of initiated certificates is independent from the percentage of malicious nodes in the network. Each node that is meeting a new neighbor, will issue a certificate for it. As there is not any prior knowledge about the new encountered neighbor, a trust threshold value, which represents the uncertainty about the trustworthiness of the subject node, is assigned to the issued certificate.

Reducing the node’s speed decreases the possibility for a node to meet a new neighboring node. Nodes with higher speed have a higher probability to find more trustworthy neighboring nodes which affect the certificate chain discovery process. This results in a greater success rate of honest certificate chains (Fig. 6.8) in networks with nodes moving with high speed, even despite of the high malicious percentage.

Next we explore the certificate chains length in different situations. Fig. 6.12 shows that the certificate chain’s length decreases in networks with nodes that are moving with high speed. We reason that more link breakage with high speed in one side and encountering of more neighbors, in other side, results in shorter discovered certificate chains.

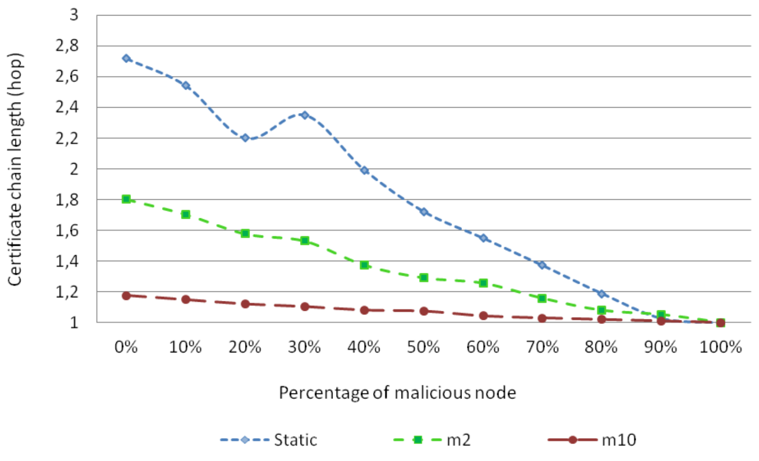


Fig. 6.12. Average of honest certificate chain length

Fig. 6.12 also demonstrates that the averaged certificate chain length decreases as the percentage of malicious nodes increases for all tested speeds. In a network with a low percentage of malicious nodes, forward ants can discover a certificate chain of a target node through more trusted neighboring nodes. However, this possibility is reduced in networks with a higher percentage of malicious nodes. Inspired from a real social life, when the number of trustworthy entities is higher in an environment, the realm of trustful acquaintance/friends is wider. However, this realm will be limited to the nearest neighbors, with more direct contact, in an environment with a higher percentage of malicious nodes. It demonstrates that trust relationships in networks with more un-trusted nodes are rather based on direct relationship than on indirect relationships or recommendations.

To investigate the performance evaluation of the proposed model for different speeds and percentages of malicious nodes, we also consider control packets transmitted in the certificate chain discovery process (Fig. 6.13). Control packets include all forward ants, backward ants [including certificates], trust update packets [consisting of punishing and rewarding packets] and select chain packets [used for the selection of the best certificate chain for the routing of data]. Only the forward ants can trigger the other kind of control packets. By increasing the percentage of malicious nodes and consequently, increasing the average of malicious neighbors, less forward ant packets have the possibility to discover the certificate chain. It means that when a node detects its neighbor is malicious, it prevents to send out forward ants toward it, for discovering a certificate chain.

Resulting from the effects of higher speeds we obtain shorter certificate chains and consequently, a smaller number of delivered certificates can be found in the certificate chain discovery phase. Therefore, higher nodes' speeds result in having less control packets in the certificate chain discovery phase.

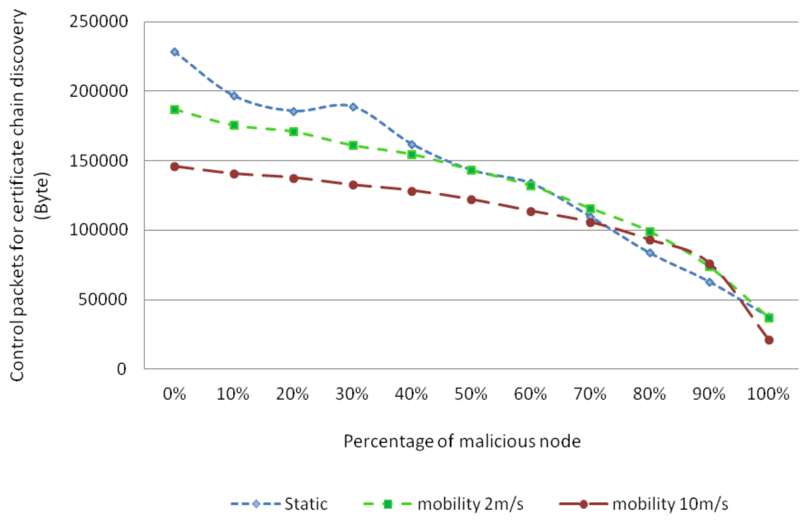


Fig. 6.13. Control packets needed for discovering of certificate chains

From low to medium percentage of malicious nodes, static networks require more control packets for certificate chain discovery phase compared to networks with node mobility. One reason for that is the length of discovered certificate chains which is longer in static networks. As certificate chains carried certificates within themselves, longer certificate chains, therefore have more certificate and consequently more burden caused by control bytes. Furthermore, as the success rate of the certificate chains in networks with lower mobility is higher, consequently it results in higher discovered certificate chains and therefore more number of control packets.

With the higher percentage of the malicious nodes in the network, as nodes with higher mobility have higher success certificate chain rate, therefore more control packets will be transmitted in higher mobility networks.

In the network with 100% malicious nodes, higher mobility (10m/s) cause faster knowing of the environment. In results source nodes, who get faster familiar with its untrustworthy neighbors, does not initiate forward ants, therefore the number of transferred control packets is lower comparing to static or low speed mobility.

6.2 Concluding Remarks

According to the obtained results, the proposed bio-inspired mechanism is suitable for self-organized authentication mechanisms without a central authority to control the security of the system. In autonomous systems, with the lacking of a supervisor, the participants themselves should learn, collaboratively and in a distributive manner, the way to adapt to the environment. In this regard ant-based approaches have the requisite conditions and are suitable candidates. Our experimental results confirm that our proposed bio-inspired self-organized authentication mechanism is adaptive for both static and mobile ad hoc networks and robust against malicious nodes in the network. The evaluation results illustrate that our proposed scheme is able to adapt well to fast changes in highly dynamic networks. It was further demonstrated that the model is robust in untrustworthy environments caused by an increase in the number of malicious nodes.

Spiritual maturation is results from either experience or pressure.
-Malak Jan Nemati

Chapter 7 Conclusion and Future Work

In this thesis we proposed an adaptive self-organized public key management authentication mechanism for mobile ad-hoc networks. The model is based on the construction and dissemination of trust. The scope of trust in our work is to be certain about the authenticity of public keys of the nodes in the network.

Different authentication mechanisms in ad-hoc networks were surveyed and divided into three classes: central authority, distributed authority and fully self-organized trust model. The hierarchical trust model (single central authority) is not suitable for these dynamic environments due to frequent topology changes and becoming a single point of attack. Distributed authority models are based on a web of trust. A public key certificate is confirmed as valid, when a certificate chain ending at a trusted certificate authority is found. However, this model cannot ensure the availability of the trusted certificate authority for nodes with high mobility, which is a problem for a security service that should be available at any time in the network. In this regards self-organized authentication mechanisms are the most appropriate mechanism for security provision in mobile ad-hoc networks.

In an autonomous key management scheme, participants in the network create their public-private key pairs. The private key is kept secret to the node, whereas the public key is published to the other nodes in the network. As there is no trusted center to confirm the authenticity of the nodes in the network, security threats emerge, while publishing public keys and binding them to the owner's identity.. In a self-organized authentication system without a trusted authority, the existence of attackers who aim at sabotaging the authentication process are not unavoidable. Therefore, trust relationships between the network participants play an important role in thwarting the threats.

The absence of a security controller forces the nodes to learn about the trustworthiness of the other entities in the network themselves. The learning process initiated from an uncertain situation, based on the result of experiences, leads to the establishment of trust relationships. Our unsupervised learning process for authenticity of the public keys is based on ant colony optimization (ACO). In providing identity trust it is important that the identity a node claims to have will be bound to a corresponding public key. With the ACO algorithm we focused on both, the construction and dissemination of trust bonded to keys. Traces of pheromone indicate the trustworthiness and authenticity of the nodes' public keys. We applied ACO aggregated with an incentive mechanism: in addition to the pheromone updating, rewarding and punishing mechanisms are utilized in order to discriminate authentic from inauthentic nodes.

The evaluation of simulation results showed that the proposed bio-inspired autonomous authentication mechanism is suitable for both, static and mobile networks. It was also demonstrated that it provides a robust behavior against malicious nodes who aim at devastating the authentication process. The model is intuitive to the nodes and mimics a real life concept. The simulation results show that nodes with mobility have a greater possibility to know new nodes, but they rely more on direct trust relationships rather than recommendations. The on-demand nature of the model causes low communication cost in terms of transferred control packets needed for the authentication process.

As malicious behavior, we consider the nodes who insert the fake certificate in the certificate chain during certificate chain discovery phase. This kind of attack can be identified through certificate chain verification by the source node. However as our future work we plan to identify the *Sybil* attack by mining the data in the certificate chains and by extracting the hidden relationships between *Sybil* nodes. The effect of mobility, whether it could be used as leverage to identify *Sybil* nodes, can be investigated. We assume that in the trust updating phase nodes are acting correctly; however, preventing attacks on trust updating can also be a future research work to improve the proposed scheme.

We also intend to further investigation of the burden of re-keying and the certificate revocation process. Another future plan, which is introduced in [99], is to investigate the scalability of our proposed scheme and expand it to the group-based autonomous authentication scheme.

Bibliography

[1] L. Rasmusson and S. Jansson, "Simulated Social Control for Secure Internet Commerce," presented at the Proceedings of the 1996 Workshop on New Security Paradigms, Lake Arrowhead, California, United States, 1996.

[2] A. Jøsang, et al., "A Survey of Trust and Reputation Systems for Online Service Provision," Decision Support Systems, vol. 43, pp. 618-644, 2007.

[3] F. Gómez Mármol, et al., "TACS, a Trust Model for P2P Networks," Wireless Personal Communications, vol. 51, pp. 153-164, 2009.

[4] S. Kamvar, et al., "The Eigentrust Algorithm for Reputation Management in P2P Networks," in Proceedings of the 12th international conference on World Wide Web (WWW '03), ACM, Budapest, Hungary, 2003, pp. 640-651.

[5] S. Marti and H. Garciamolina, "Taxonomy of Trust: Categorizing P2P reputation systems," Computer Networks, vol. 50, pp. 472-484, 2006.

[6] S. Buchegger and J. Le Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks," in Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems, 2004.

[7] P. Michiardi and R. Molva, "Core: a Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," presented at the Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security, 2002.

[8] A. Boukerch, et al., "Trust-based Security for Wireless Ad Hoc and Sensor Networks," Computer Communication, vol. 30, pp. 2413-2427, 2007.

[9] S. Ganeriwal and M. B. Srivastava, "Reputation-based Framework for High Integrity Sensor Networks," presented at the ACM Transactions on Sensor Networks (TOSN), Washington DC, USA, 2004.

[10] A. A. Pirzada and C. McDonald, "Establishing Trust in Pure Ad Hoc Networks," presented at the Proceedings of the 27th Australasian conference on Computer science - Volume 26, Dunedin, New Zealand, 2004.

[11] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," IEEE Network: special issue on network security, vol. 3, pp. 24–30, 1999.

[12] A. J. Menezes, et al., Handbook of Applied Cryptography. Florida, USA: CRC Press, 1997.

[13] H. Sarosh and J. Brooke, "Authentication Mechanisms for Mobile Ad-Hoc Networks and Resistance to Sybil Attack," in Proceedings of the second International Conference on Emerging Security Information, Systems and Technologies, 2008, pp. 120-126.

[14] P. Zimmermann, The Official PGP User's Guide: MIT Press, 1995.

[15] J. Brainard, et al., "Fourth-factor Authentication: Somebody You Know," presented at the Proceedings of the 13th ACM conference on Computer and communications security, Alexandria, Virginia, USA, 2006.

[16] S. Capkun, et al., "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," IEEE Transactions on Mobile Computing, vol. 2, pp. 52-64, 2003.

[17] R. Li, et al., "On-Demand Public-Key Management for Mobile Ad Hoc Networks: Research Articles," Wireless Communications & Mobile Computing - Wireless Network Security, vol. 6, pp. 295-306, 2006.

[18] H. Dahshan and J. Irvine, "A Robust Self-organized Public Key Management for Mobile Ad Hoc Networks," Security and Communication Networks, vol. 3, pp. 16-30, 2010.

[19] O. Cordon, et al., "A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends," Mathware & Soft Computing vol. 9, pp. 141-175, 2002.

[20] D. Spiewak and T. Engel, "Applying Trust in Mobile and Wireless Networks," in Mobile and Wireless Network Security and Privacy, K. Makki, et al., Eds., ed: Springer US, 2007, pp. 39 - 67.

[21] J. R. Douceur, "The Sybil Attack," presented at the Revised Papers from the First International Workshop on Peer-to-Peer Systems, 2002.

[22] R. Song, et al., Trust in E-services: Technologies, Practices and Challenges, Chapter 2: IGI Publishing, 2007.

[23] A. Josang, "The Right Type of Trust for Distributed Systems," presented at the Proceedings of the Workshop on New Security Paradigms, Lake Arrowhead, California, United States, 1996.

[24] A. Jøsang, et al., "Can We Manage Trust?," in Proceedings of the 3th International Conference on Trust Management (iTrust), Versailles, 2005.

[25] P. Lamsal, "Understanding Trust and Security," Department of Computer Science. University of Helsinki Finland,2001.

[26] D. Gambetta, "Can We Trust Trust?," in Trust: Making and Breaking Cooperative Relations,Department of Sociology, University of Oxford, chapter 13, 2000, pp. 213-237.

[27] S. Marsh, "Formalising Trust as a Computational Concept," Phd Thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[28] P. Zimmermann, "The Official PGP User’s Guide," MIT Press, 1995.

[29] L. Capra, "Engineering Human Trust in Mobile System Collaborations," presented at the Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering, USA, 2004.

[30] E. Cayirci and C. Rong, Security in Wireless Ad Hoc and Sensor Networks, 1 ed.: A John Wiley and Sons, Ltd, Publication, 2009.

[31] Y. Han, et al., "A Survey of Trust and Reputation Management Systems in Wireless Communications," Proceedings of the IEEE, vol. 98, pp. 1755-1772, 2010.

[32] S. Ganeriwal, et al., "Reputation-based Framework for High Integrity Sensor Networks," ACM Transactions on Sensor Networks (TOSN), vol. 4, pp. 1-37, 2008.

[33] A. Urpi, et al., "Modeling Cooperation in Mobile Ad Hoc Networks: a Formal Description of Selfishness," in Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, France 2003, pp. 3-5.

[34] S. Buchegger and J.-Y. L. Boudec, "Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks)," in 3rd ACM international symposium on Mobile ad hoc networking & computing, Lausanne, Switzerland, 2002, pp. 226-236.

[35] Q. He, et al., "SORI : a Secure and Objective Reputation-based Incentive Scheme for Ad Hoc Networks," in IEEE Wireless Communications and Networking Conference, 2004, pp. 825 - 830.

[36] T. Jiang and J. S. Baras, "Ant-Based Adaptive Trust Evidence Distribution in MANET," presented at the Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04) - Volume 7, 2004.

[37] D. S. Dawoud, et al., "Trust Establishment in Mobile Ad Hoc Networks: Key Management," in Mobile Ad hoc Networks: Application, InTech, X. Wang, Ed., ed, 2011.

[38] N. Aboudagga, et al., "Authentication Protocols for Ad Hoc Networks: Taxonomy and Research Issues," presented at the Proceedings of the 1st ACM international workshop on Quality of service and security in wireless and mobile networks, Montreal, Quebec, Canada, 2005.

[39] A. M. W. Hegland, E. Mjolsnes, S.F. Rong, C. Kure, O. Spilling, P. . (2006) A Survey of Key Management in Ad Hoc Networks. Communications Surveys & Tutorials, IEEE. 48-66.

[40] W. Stallings, Cryptography and Network Security: Principles and Practice, (2nd ed.) ed.: Prentice-Hall, 1999.

[41] RFC3280. http://www.ipa.jp/security/rfc/RFC3280-00EN.html.

[42] R. R. S. Verma, et al., "NTM - Progressive Trust Negotiation in Ad Hoc Networks," in Proceeding of Joint IEI/IEE Symposium on Telecommunication Systems Research, Dublin, Ireland, 2001.

[43] L. Zhou and Z. J. A Haas, "Securing ad hoc networks," Network, IEEE, vol. 13, pp. 24 -30, 1999.

[44] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, pp. 612-613, 1979.

[45] A. Herzberg, et al., "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," Lecture Notes in Computer Science, vol. 963, pp. 339-352, 1995.

[46] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," in 2nd Annual PKI Research Workshop Program (PKI 03), 2003.

[47] S. Basagni, et al., "Secure Pebblenets," presented at the Mobihoc 01 Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, Long Beach, CA, USA, 2001.

[48] A. Abdul-Rahman, "The PGP Trust Model," EDI-Forum: The Journal of Electronic Commerce, vol. 10, pp. 27-31, 1997.

[49] D. Balfanz, et al., "Talking to strangers: Authentication in ad-hoc wireless networks," 2002.

[50] Hubaux J.-P., et al., "The Quest for Security in Mobile Ad Hoc Networks

" in 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, CA, USA, 2001.

[51] S. Capkun, et al., "Mobility Helps Peer-to-Peer Security," IEEE Transactions on Mobile Computing, vol. 5, pp. 43-51, 2006.

[52] S. Capkun, "Location verification and key management in wireless networks," in EPFL, ed, 2004.

[53] P. Memarmoshrefi, et al., "Bio-inspired Self-organized Public Key Authentication Mechanism for Mobile Ad-hoc Networks," Bio-Inspired Models of Network, Information, and Computing Systems pp. 375-386, 2012.

[54] P. Memarmoshrefi, et al., "Autonomous Ant-based Public Key Authentication Mechanism for Mobile Ad Hoc Networks," to appear in International Journal of Autonomous and Adaptive Communications Systems (IJAACS), 2012.

[55] A. Jøsang and J. Golbeck, "Challenges for Robust of Trust and Reputation Systems," in Proceedings of the 5th International Workshop on Security and Trust Management (STM 2009), Saint Malo, France, 2009.

[56] A. Förster, "Machine Learning Techniques Applied to Wireless Ad-Hoc Networks: Guide and Survey," in Proceedings of the The 3th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, 2007.

[57] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction: The MIT Press, 1998.

[58] L. P. Kaelbling, et al., "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996.

[59] J. Kennedy and R. Eberhart, Swarm Intelligence: Morgan Kaufmann, 2001.

[60] C. Tham, et al., "Mobile Agents Based Routing Protocol for Mobile Ad Hoc Networks," presented at the Global Telecommunications Conference, GLOBECOM '02., 2002.

[61] S. Koenig, "Agent-centered Search," AI Magazine vol. 22, pp. 109-131, 2001.

[62] R. E. Korf, "Real-time heuristic search," Journal Artificial Intelligence, Elsevier Science Publishers Ltd. Essex, UK, vol. 42, pp. 189-211, 1990.

[63] M. Rossi, et al., "Statistically Assisted Routing Algorithms (SARA) for Hop Count Based Forwarding in Wireless Sensor Networks," Wireless Networks, vol. 14, 2006.

[64] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach: Prentice Hall Series in Artificial Intelligence, 2003.

[65] R. Rojas, Neural Networks - A Systematic Introduction: Book, Springer-Verlag, Berlin, New-York, 1996.

[66] T. Mitchell, Machine Learning: McGraw-Hill, 1997.

[67] M. Dorigo, et al., "Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique," IEEE Computational Intelligence Magazine, vol. 1, pp. 28-39, 2006.

[68] M. Dorigo, "Optimization, Learning and Natural Algorithms.," Ph.D. Thesis Politecnico di Milano, Italy, 1992.

[69] M. Dorigo and G. D. Caro, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, pp. 137-172, 1999.

[70] S. Luke, Essentials of Metaheuristics: A Set of Undergraduate Lecture Notes, 2009.

[71] M. Dorigo and G. D. Caro, "The Ant Colony Optimization Meta-heuristic," in New Ideas in Optimization, ed: McGraw-Hill Ltd., UK, 1999, pp. 11-32.

[72] M. Dorigo, et al., "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 26, pp. 29-41, 1996.

[73] M. Dorigo, et al., "Positive Feedback as a Search Strategy," Technical Report, Dipartimento di Elettronica, Politecnico di Milano, Italy1991.

[74] L. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem," in International Conference on Machine Learning, 1995, pp. 252-260.

[75] M. Dorigo and L. M. Gambardella, "Ant Colonies for the Travelling Salesman Problem " in Biosystems, 1997, pp. 73-81.

[76] M. Dorigo and L. M. Gambardella, "Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Transactions on Evolutionary Computation vol. 1, pp. 53-66, 1997.

[77] L. Gambardella and M. Dorigo, "Solving Symmetric and Asymmetric TSPs by Ant Colonies," in International Conference on Evolutionary Computation, 1996, pp. 622-627.

[78] T. Stuetzle and H. H. Hoos, "MAX-MIN Ant system," Future Generation Computer Systems, vol. 16, pp. 889-914, 2000.

[79] B. Bullnheimer, et al., "A New Rank Based Version of the Ant System," Central European Journal of Operations Research, vol. 7, pp. 25-38, 1999.

[80] V. Maniezzo, "Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem," INFORMS Journal on Computing, vol. 11, pp. 358-369, 1999.

[81] O. Cordón, et al., "A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System," in Proceedings of ANTS 2000 - From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms, 2000, pp. 22-29.

[82] C. Blum and M. Dorigo, "The Hyper-Cube Framework for Ant Colony Optimization," IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), vol. 34, pp. 1161-1172, 2004.

[83] L. Bianchi, et al., "An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem," presented at the Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, 2002.

[84] K. Socha, et al., "Ant Algorithms for the University Course Timetabling Problem With Regard to the State-of-the-Art," presented at the Proceeding 3th European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003), Essex, UK, 2003.

[85] D. Costa and A. Hertz, "Ants Can Colour Graphs," The Journal of the Operational Research Society, vol. 48, pp. 295-305, 1997.

[86] D. Merkle, et al., "Ant Colony Optimization for Resource-Constrained Project Scheduling," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 333-346, 2002.

[87] C. Blum, "Beam-ACO: Hybridizing Ant Colony Optimization with Beam Search: an Application to Open Shop Scheduling," Computers and Operations Research, vol. 32, pp. 1565-1591, 2005.

[88] G. D. Caro, "Ant Colony Optimization and Its Application to Adaptive Routing in Telecommunication Networks," PhD Thesis, Université Libre de Bruxelles (ULB), Brussels, 2004.

[89] G. D. Caro and M. Dorigo, "Antnet: Distributed Stigmergetic Control For Communications Networks," Journal of Artificial Intelligence Research, vol. 9, pp. 317-365, 1998.

[90] W. Wang, et al., "Ant-based Reputation Evidence Distribution in P2P Networks," presented at the Proceedings of the 5th International Conference on Grid and Cooperative Computing, 2006.

[91] G. D. Caro, et al., "AntHocNet: An Adaptive Nature-inspired Algorithm for Routing in Mobile Ad Hoc Networks," European Transactions on Telecommunications, vol. 16, pp. 443--455, 2005.

[92] F. Ducatelle, "Adaptive Routing in Ad Hoc Wireless Multi-hop Networks," PhD Thesis, Faculty of Informatics, Universita della Svizzera Italiana, 2007.

[93] M. K. Reiter and S. G. Stubblebine, "Authentication Metric Analysis and Design," ACM Transactions on Information and System Security (TISSEC), vol. 2, pp. 138-158, 1999.

[94] M. Gasser, et al., "The Digital Distributed System Security Architecture," 1989.

[95] J. J. Tardo and K. Alagappan, "SPX: Global Authentication Using Public Key Certificates," presented at the IEEE Symposium on Security and Privacy, 1991.

[96] S. T. Kent, "Internet Privacy Enhanced Mail," Communications of the ACM - Special issue on internetworking, vol. 36, pp. 48-60, 1993.

[97] M. K. Reiter and S. G. Stubblebine, "Resilient Authentication Using Path Independence," IEEE Transactions on Computers, vol. 47, pp. 1351-1362, 1998.

[98] A. Jabas, "MANET Mining: Mining Association Rules," in Mobile Ad-Hoc Networks: Applications, InTech X. Wang, Ed., ed, 2011.

[99] P. Memarmoshrefi, et al., "Autonomous Group-based Authentication Mechanism in Mobile Ad-hoc Networks," presented at the In Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE 11th International Conference Liverpool, UK, 2012.