The Quest for Security in Mobile Ad Hoc Networks*

Jean-Pierre Hubaux
Institute for Computer Communications and Applications
Swiss Federal Institute of Technology – Lausanne
CH-1015, Switzerland
Jean-Pierre.Hubaux@epfl.ch

Levente Buttyán
Institute for Computer Communications and Applications
Swiss Federal Institute of Technology – Lausanne
CH-1015, Switzerland
Levente.Buttyan@epfl.ch

Srđan Čapkun
Institute for Computer Communications and Applications
Swiss Federal Institute of Technology – Lausanne
CH-1015, Switzerland
Srdan.Capkun@epfl.ch

ABSTRACT

So far, research on mobile ad hoc networks has been focused primarily on routing issues. Security, on the other hand, has been given a lower priority. This paper provides an overview of security problems for mobile ad hoc networks, distinguishing the threats on basic mechanisms and on security mechanisms. It then describes our solution to protect the security mechanisms. The original features of this solution include that (i) it is fully decentralized and (ii) all nodes are assigned equivalent roles.

1. INTRODUCTION

The recent history of the Internet and of cellular networks has shown that if security of a given network architecture is not properly designed from the very beginning, then the security breaches will be exploited by malicious users. Moreover, introducing or reinforcing security mechanisms *a posteriori* can be a very painful and expensive process. Security in mobile ad hoc networks is particularly difficult to achieve, notably because of the vulnerability of the links, the limited physical protection of each of the nodes, the sporadic nature of connectivity, the dynamically changing topology, the absence of a certification authority, and the lack of a centralized monitoring or management point.

Clearly, security requirements depend very much on the kind of mission for which the mobile ad hoc network has been conceived, and the environment in which it has to operate. For example, a military mobile ad hoc network certainly will have very stringent requirements in terms of confidentiality and resistance to denial of service attacks. Mechanisms to encourage cooperation between nodes (as presented later) can be highly desirable in a civilian context, whereas they do not make much sense in their military counterpart. Moreover, anonymity will usually be desirable in both military and civilian contexts, but with different flavors: in the case of the battlefield, it is important to hide the location of the headquarters, whereas in a commercial scenario, a consumer may wish to protect his privacy with respect to a given service provider or merchant. Another example is "zero configuration" networks (www.zeroconf.org), where nodes must be able to authenticate each other even in the absence of any infrastructure. Yet another example is "spontaneous networking" [12], where security associations can leverage on the trust relationships between the people involved. Finally, a network of sensors will generally have security requirements that are quite different from ad hoc networks comprised of personal communication devices [21].

Nevertheless, security requirements and mechanisms exhibit interesting commonalities across most mobile ad hoc networks, which we are now going to present and discuss. A property that we will mention several times is *self-organization*, namely the ability of a mobile ad hoc network to work without any external management or configuration. Although not all mobile ad hoc networks are self-organized, this property is assumed in most of the ongoing research projects.

Security can be perceived and implemented at different levels. In particular, it is possible to embed security mechanisms in the application. This paper will not consider this kind of security; rather, it will focus on the security mechanisms which are related to the very nature of the kind of network under scrutiny.

The rest of the paper is organized in the following way. In Section 2, we will refine the identification of the threats. In Section 3, we will see how the basic mechanisms of such a network can be protected. In Section 4, we will address the difficult question of protecting the security mechanisms. In Section 5, we will present our solution to the latter problem. We will conclude the paper in Section 6.

When appropriate, we will refer to a research program we have recently started, called *Terminodes* [1, 14, 5], focused on self-organized mobile ad hoc networks.

2. THREATS

In this section, we will describe the most important threats that mobile ad hoc networks have to face. One can distinguish essentially two levels of attack:

- Attacks on the *basic mechanisms* of the ad hoc network, such as routing. Prevention of these attacks requires security mechanisms that are often based on cryptographic algorithms.
- Attacks on the *security mechanisms* and notably on the *key management mechanisms.* Key management is certainly not a problem limited to ad hoc networks. However, because of the peculiarities of ad hoc networks, its solution requires specific attention.

2.1 Vulnerabilities of the basic mechanisms

Unlike nodes of conventional (wireline) networks, nodes of ad hoc networks cannot be assumed to be secured in locked cabinets. Therefore, they risk being **captured and compromised**. The terminals of cellular networks, for example, have been frequently stolen or tampered with by rogue users, entailing losses to the order of hundreds of millions of dollars for the operators.

As all communications are performed **over the air**, ad hoc networks are vulnerable to attacks ranging from eavesdropping to active interference.

Another problem related to the previous one is that the algorithms are assumed to be **cooperative**. For example, in a MAC layer, nodes are expected to cooperate. In a contention-based mechanism, nodes must follow the predefined rules to avoid collisions or recover from them. In a contention-free mechanism (which is better suited to ad hoc networks), each node must obtain an agreement from all the others for an exclusive use of the channel resource. In both cases, if a node does not respect the rules, the allocation of the communication channel will be unfair and the performance of the network can be severely affected.

Routing mechanisms are more vulnerable in ad hoc networks than in conventional networks because in ad hoc networks each device acts as a relay. This means, for example, that an adversary who hijacks an ad hoc node could paralyze the entire network by disseminating false routing information. A less dramatic but more subtle malicious behavior is node *selfishness*: some nodes may be tempted to not relay packets (e.g., in order to save their own battery) [18].

Moreover, weaknesses in the protocols can be exploited to perform malicious **neighbor discovery**. For example, researchers have recently shown how this kind of attack can be performed against a Bluetooth device [15].

2.2 Vulnerabilities of the security mechanisms

In virtually any network, the fundamental security mechanisms require that the users make use of appropriate cryptographic keys. As mentioned in [20], the goal of a good cryptographic design is to reduce complex problems to the proper management and safe-keeping of a small number of cryptographic keys. This objective is difficult to accomplish in an ad hoc network (in which the different nodes move around and in which connectivity is not guaranteed).

Examples of attacks against the security mechanisms are the following: public keys can be maliciously replaced; some keys can be compromised; if there is a (distributed) trusted server, it can fall under the control of a malicious party. These threats are not specific to ad hoc networks, but the solutions have to take into account the peculiarities of ad hoc networks.

3. PROTECTION OF THE BASIC MECHANISMS

In this section, we will consider a certain number of solutions that can be deployed in order to thwart the attacks mentioned in Subsection 2.1. We will not address the protection of the radio interface (e.g., prevention of eavesdropping and jamming), as this issue is not really specific to mobile ad hoc networks. This problem has been intensively researched for virtually all wireless networks and many solutions have been proposed and deployed, such as spread spectrum communication and frequency hopping [11].

3.1 Tamper resistance

As a device is at risk of being captured and hijacked, it must be protected in some way. The conventional solution consists in protecting the device (or part of it) by implementing it in tamper resistant hardware (for a critical assessment of tamper resistant devices, see [2, 22]). A first option would consist in embedding the cryptographic information (the secret key, typically) in a smart card, which could be plugged and removed at will into and from the node itself; the SIM card based solution of GSM works according to this principle. The advantage of a removable card is that it allows a user to change devices while keeping her own private data. An additional advantage is that all the sensitive information is protected by the smart card. There are also drawbacks: by its nature, the smart card has no direct input/output capabilities towards the user, it does not have its own power supply neither does it have its own clock. This makes smart cards vulnerable to attacks mounted from a compromised device in which they are plugged in.

An additional need is to protect the networking mechanisms embedded in the node (e.g., routing). A solution is to store the related software in a smart card. If the smart card is not powerful enough, a possibility consists in making use of security processors, which comprise a processor, some memory, and appropriate tamper detection circuitry [2]. This can provide more security than the smart card. There is still a problem, however: a software package such as routing has to be upgraded from time to time. Therefore, there must be a mechanism by which the operating system can check that a new version of software is a legitimate one.

A related issue is *system imprinting*: at initialization, a system must be told in one way or another to whom and how it has to obey (namely who are its *users*, what is the identity or the category of the other devices it is entitled to communicate with, what are the access rights, etc). The solutions will very much depend on the *raison d'être* of the mobile ad hoc network. For an entertaining and thought-provoking discussion of this issue, see [25]. This paper also discusses among others the battery exhaustion attack.

A related problem is to know whom to trust. If the device is equipped with a tamper resistant security module, how do we make sure that the latter has not been replaced by a faked one? Even if it is the legitimate one, to what extent do we trust the manufacturer? Clearly, the manufacturer must be regularly scrutinized by an appropriate authority. But here again, do we trust this authority?

3.2 Routing-based mechanisms

In mobile ad hoc networks, a lot of research has been devoted to routing algorithms. However, in most cases, the nodes are assumed to be cooperative. Initial work on mitigating routing misbehavior in mobile ad hoc networks is proposed in [18]. In this paper, the authors consider the case in which some malicious nodes agree to forward packets but fail to do so. In order to cope with this problem, they propose two mechanisms: a *watchdog*, in charge of identifying the misbehaving nodes, and a *pathrater*, in charge of defining the best route circumventing these nodes.

The paper shows that these two mechanisms make it possible to maintain the total throughput of the network at an acceptable level, even in the presence of a high amount of misbehaving nodes (e.g., 40%). However, the operation of the watchdog is based on an assumption which is not always true (as reckoned by the authors): the *promiscuous* mode of the wireless interface. Another problem is that the selfishness of the nodes does not seem to be castigated; on the contrary, by the combination of the watchdog and the pathrater, the misbehaving nodes will not be bothered by the transit traffic, while still enjoying the possibility to generate and to receive traffic. The proposed mechanisms could be enriched in such a way that a misbehaving node would be locked out by its neighbors. There would still be two problems, however: (i) the lockout mechanism could be exploited to mount denial-of-service attacks and (ii) a locked out node could simply move away, in an area where his misbehavior has not yet been reported.

On the other hand, as pointed out in [28], certain properties of ad hoc networks can be exploited to achieve secure routing. Routing protocols of ad hoc networks have to cope with outdated routing information to accomodate the dynamically changing topology. False routing information generated by compromised nodes could, to some extent, be considered outdated information. As long as the number of correct nodes remains high enough, the routing protocol should be able to find routes that circumvent the compromised nodes. As routing protocols can discover multiple routes, nodes can switch to an alternative route when the primary route appears to have failed.

Multiple routes may require that the same packet is sent several times, leading to a wastage of resources. However, an appropriate coding strategy can avoid message retransmission. The basic idea is to transmit redundant information through additional routes for error correction and detection. Two examples of this approach are *diversity coding* [4] and *multiple description subband coding* [14].

An attacker can also try to modify the content of the routing table. The simplest way to thwart such an attack is to avoid routing tables, and to base packet forwarding on geographic information [14, 5]. However, this requires that each of the nodes is aware of its own geographic position and is able to share it with others, hence creating other kinds of vulnerabilities.

To conclude this subsection, it is worth mentioning that researchers are also exploring the application of *intrusion detection techniques* to the protection of mobile ad hoc networks [27]. Moreover, prevention of traffic analysis has also been considered [16].

3.3 Neighborhood

Attacks can be based on the protocols between neighbors, such as the hello protocol. By this technique, an attacker can force a victim node to unveil private data, such as its identity. In fact, even in the much simpler case of cellular networks, in which users can rely on their home network operator to protect their privacy, many solutions have been proposed, but the problem is not yet really solved [6].

In the framework of Bluetooth, the authors of [15] show how the activity of a victim can be observed by a set of devices installed in strategic places by an attacker. In the same paper, they suggest a solution based on pseudonyms: if the identity of a device changes for each session, then it becomes much more difficult for an intruder to trace its location and its activities. However, this induces a higher complexity of the addressing schemes.

3.4 Service enforcement

If an ad hoc network is self-organized, service availability is a major requirement. There are two issues: First, end-users must be given incentives to cooperate (and notably to relay packets for the benefit of other users). Second, they must be discouraged from overloading the network. In mobile ad hoc networks proposed and implemented so far, these two aspects were irrelevant because of the small size of the network and the emergency situations in which they were expected to be deployed. Moreover, the nodes belonged to the same authority and shared the same goals.

In our research program [1, 14, 5], we have removed these assumptions. Relying on the observation that sophisticated transactions are usually based on some form of currency, we have devised a mechanism based on a virtual currency called a "nuglet" [7]: nodes remunerate each other for the services they provide to each other. In this way, a node can make use of the network only if it also contributes to the benefit of the community. In the context of the Internet, a similar example, aimed at fostering Web publishing by micro-transactions, is Mojo Nation (www.mojonation.net).

We have explored the application of nuglets to stimulate the provision of the packet forwarding service. We have proposed two charging models: the *Packet Purse Model*, in which the source is charged, and the *Packet Trade Model*, in which the destination is charged. Figure 1 illustrates the operation of the Packet Purse Model. For a more detailed description of the models, we refer to [8].

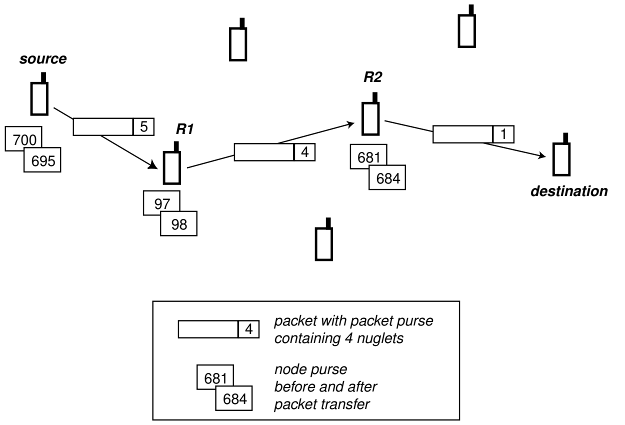

Figure 1: Virtual currency: the nuglets - The source puts 5 nuglets in the packet purse. The first relaying node $R1$ charges 1 nuglet, which is taken from the packet purse. The second relaying node $R2$ charges 3 nuglets. This higher greediness can be motivated, for instance, by the fact that $R2$ has to face a higher energy expenditure to relay the packet (typically if the distance to the next hop is higher).

The protection of the models against misuse could be based on a tamper resistant hardware module in each device. This module would manage the nuglets of the node in such a way that the node cannot increase its stock of nuglets in an illegitimate way. In addition, this module would also be used to provide cryptographic protection for the packet purses. A protocol that could be used for this purpose is described in [8].

4. PROTECTING THE SECURITY MECHANISMS

As we have mentioned in Subsection 2.2, protecting the security mechanisms in an ad hoc network is a major challenge. We will focus here on what is arguably the most critical and complex issue, namely key establishment. As mentioned in [20], key establishment can be realized by key transport or key agreement. In key transport, one party creates or otherwise obtains a secret value, and securely transfers it to the other(s). In key agreement, a shared key is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) in such a way that no party can predetermine the resulting value. Both approaches can be based on symmetric or asymmetric techniques, and there are a number of well defined protocols to achieve this goal [20].

The identification of the appropriate solution in an ad hoc network will depend on a number of criteria, such as: What are the expected security functions to be implemented (confidentiality, integrity, authentication,...)? Is there an authority, or are there several authority domains? If there is an authority, what is its role in the initialization phase (e.g., can the authority install appropriate cryptographic material in each node before usage)? If there are several authority domains, what is the level of trust between them, and how do they build up a trust relationship? If there is a trusted server (or several ones), is it accessible on-line? Is there an upper bound on the number of users, known a priori? Are trust relationships between any two users publicly known or available? Is key establishment limited to two parties, or is also multi-party key establishment to be considered? Last but not least, what exactly is the key life cycle; in particular, is key revocation to be considered?

For lack of space, we cannot discuss each of these criteria. We will assume that there is no authority (or, in an equivalent way, that each node is its own authority domain); we will also assume that there is no fixed server.

Asymmetric key cryptography is an appropriate concept for this case[1], because it does not require online trusted servers. However, there is a drawback related to key revocation. The usual scalable mechanism to achieve this is that an authority maintains a list of revoked keys on a server, a solution clearly not adapted to our problem. An alternative would be to request the public key directly from its owner. But this would have to be realized for each new interaction in a secure way, and therefore the expected benefits of the asymmetric technique would be lost.

In spite of this drawback, in the rest of our discussion, we will focus on key establishment using asymmetric key cryptography. In such a system, each node has a public/private key pair. Public keys can be distributed to other nodes, while private keys should be kept confidential to individual nodes. A crucial problem for a given node A is how to obtain the authentic public key of a node B. The most important threat is an *intruder-in-the-middle* attack [20]. We will discuss this issue in Section 5.

A way to use asymmetric key cryptosystems for the transport of the symmetric keys is to encrypt a symmetric key generated by one party with the public key of the other party.

As for key agreement, Diffie and Hellman have suggested a technique of *key exchange* [9] based on asymmetric key cryptography. In this approach, two parties wanting to communicate securely begin their interaction by exchanging (appropriately constructed) random values, from which both compute locally the same key. The basic version provides protection in the form of secrecy of the resulting key from passive adversaries. In order to thwart active attacks, such as the intruder-in-the-middle attack, several proposals have been made. Three prominent examples thereof are the *Station-to-Station protocol* [10], the *MTI key agreement protocol* [19] and *key agreement using self-certifying keys* [13]. They propose a way to bind the exchanged random values to the identities of the parties. The problem is that this requires involving a trusted party (typically when a given party is initialized for the very first time). In a self-organized mobile ad hoc network, this would require that this role is played by a unique authority (e.g., a single manufacturer of the security module of each node); we consider that this assumption is too restrictive for our case.

[1] In practice, for efficiency reasons, symmetric key schemes are used to secure further communication after the nodes have authenticated each other and established a secret symmetric key using asymmetric key cryptography.

Identity-based cryptosystems [24] are a way to circumvent the problem of binding public keys to identities. In these systems, an entity's public identification information (unique name) plays the role of its public key, thus avoiding the need for users to exchange (and certify in some way) their public keys; a further advantage is that public directories of key certificates do not need to be kept. However, the drawback is that a trusted authority is needed for the establishment of the private keys of the users. In theory, this trusted authority is required only at system set up. But in practice, its activation would still be necessary in case e.g. a private key gets lost or compromised. For this reason, this approach does not seem to be appropriate for our case.

There are essentially three families of approaches for eliminating a centralized certification authority in a mobile ad hoc network. The first consists in *emulating* a conventional certification authority by distributing it on several nodes; the second consists in a totally distributed solution, where nodes have to authenticate each other by setting up an appropriate context; we will describe these first two options hereafter. The third one is our proposal; it is based on a self-organized public-key infrastructure. We will discuss it in Section 5.

4.1 Emulation of a certification authority

In [28], the authors propose a key management service, distributed over a certain number of nodes called *servers*. The service, as a whole, has a public/private pair K/k. The public key K is known to all nodes in the network, whereas the private key k is divided into n shares $s_1, s_2, ...s_n$, one share for each server. Each server i also has a public/private key K_i/k_i, and knows the public keys of all nodes. The n servers are chosen arbitrarily among the nodes of the network.

In order to protect itself against the potential compromise of some of the n servers, the system uses *threshold cryptography*. An $(n, t+1)$ threshold cryptography scheme ($n \geq 3t+1$) allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature) so that any $t+1$ parties can perform this operation jointly, whereas it is infeasible for at most t parties to do so even by collusion.

Mobile adversaries (specifically, attackers that temporarily compromise a server and then move on to the next victim) could progressively compromise all the servers. In order to thwart these kinds of attack, proactive schemes are proposed. They make use of *share refreshing*, which enables servers to compute new shares from old ones in collaboration, without disclosing the service private key to any server. The new shares constitute a new $(n, t+1)$ sharing of the service's private key.

This mechanism can be very robust against sophisticated attacks, and is therefore well suited for military applications. However, it requires that a subset of the nodes (the servers) play a specific role at a given point in time, an undesirable requirement for self-organized civilian networks, in which each user is expected to behave in a "selfish" way.

The next subsection discusses ways to eliminate this constraint, at the expense of a strong locality assumption.

4.2 Key agreement

As mentioned in [3], nodes willing to establish a secure session must share a *prior context*. The scenario considered by the authors is a small group of people at a conference coming together in a room for an ad hoc meeting and willing to set up a wireless network session among their laptop computers for the duration of the meeting. It is assumed that they do not have access to public key infrastructure or third party key management service. The proposed solution is the following: a fresh password is chosen and shared among those present in the room (e.g., by writing it on a blackboard). However, it would be a mistake to use this password directly as the key, as the protocol would then be vulnerable to *dictionary attacks* [20]. Therefore, the authors propose to make use of *password-authenticated key exchange* by which they derive a strong shared key starting from only a weak shared key. This proposal only works if the parties can share a password by being physically present in the same room.

5. SELF-ORGANIZED PUBLIC-KEY INFRASTRUCTURE

The problem of public-key distribution in general can be summarized in the following question: How can a user u obtain the authentic public key of another user v in the presence of an active attacker? The most well-known approach to solve this problem is based on public-key certificates. A public-key certificate is a data structure in which a public key is bound to an identity (and possibly to some other attributes) by the digital signature of the issuer of the certificate. When user u wants to obtain the authentic public key of user v, it acquires a chain of public-key certificates such that the first certificate of the chain can directly be verified by u using a public key that u holds and trusts; each remaining certificate can be verified using the public key in the previous certificate of the chain; and the last certificate contains the public key of the target v. It is assumed that u trusts the issuer of each certificate in the chain.

In many of the known certificate based systems (e.g., in Privacy Enhanced Mail [17]) public-key certificates are issued by trusted third parties, called Certification Authorities. This is not a self-organized approach for obvious reasons, and therefore, it is not appropriate for self-organized mobile ad hoc networks. In other systems (e.g., in Pretty Good Privacy (PGP) [29]), certificates are issued by the users themselves; however, the distribution of certificates is based on publicly accessible certificate directories that reside on centrally managed servers. For this reason, this approach is not fully self-organized either.

We propose a new public-key distribution system suitable for self-organized mobile ad hoc networks, which is similar to PGP in the sense that public-key certificates are issued by the users. However, as opposed to PGP, we do not rely on certificate directories for the distribution of certificates. Instead, in our system, certificates are stored and distributed by the users. Each user maintains a local certificate repository that contains a limited number of certificates selected by the user according to some algorithm. When user u wants to obtain the public key of user v, they merge their local certificate repositories, and u tries to find an appropriate certificate chain from u to v in the merged repository. We present algorithms to construct local certificate repositories such that, if used by each user, any pair of users can find certificate chains to each other in their merged repository with high probability even if the size of the local repositories is small compared to the total number of users of the system. This means that our approach is scalable; however, it provides only probabilistic guarantees.

5.1 Model and framework

We assume that if a user u believes that a given public key belongs to a given user v, then u issues a public-key certificate to v. Furthermore, we assume that users are honest and do not issue false certificates. We briefly address the problem of dishonest users in Subsection 5.4.

We model the relationships between users represented by the public-key certificates as a directed graph $G(V, E)$, where V and E stand for the set of vertices and the set of edges, respectively. We call this graph the *trust graph*. The vertices of the trust graph represent users and the edges represent public-key certificates. More precisely, there is a directed edge from vertex u to vertex v if user u issued a public-key certificate to user v.

A certificate chain from user u to user v is represented by a directed path from vertex u to vertex v in G. For any directed graph H, if two vertices u and v are in H, and there is a directed path from u to v in H, then we say that *v is reachable from u in H* and we denote this by $u \rightsquigarrow_H v$. Thus, the existence of a certificate chain from user u to user v means that vertex v is reachable from vertex u in G (i.e., $u \rightsquigarrow_G v$).

As we mentioned before, in our public-key distribution system, each user maintains a local repository of public-key certificates. This repository has two parts. First, each user stores the certificates that she issued. This is needed in order to store all the certificates issued in the system in a decentralized way. Second, each user stores a set of selected certificates issued by other users in the system. In terms of our model, this means that each user u stores the outgoing edges (with the corresponding vertices) from vertex u and an additional set of selected edges (with the corresponding vertices) of the trust graph. We refer to the set of selected edges (and vertices) as the subgraph that belongs to u.

When user u wants to verify the public key of user v, u and v merge their repositories of selected certificates, and u tries to find an appropriate certificate chain from u to v in the merged repository. In the model, u and v merge their subgraphs, and u tries to find a path from vertex u to vertex v in the merged subgraph. An example is shown in Figure 2.

We assume that each user uses the same subgraph selection algorithm $\mathcal{A}$ to build her subgraph. When $\mathcal{A}$ is executed on G by user u, it results in a subgraph that we denote by $S_{\mathcal{A}}(G, u)$. The union of the subgraphs $S_{\mathcal{A}}(G, u)$ of user u and $S_{\mathcal{A}}(G, v)$ of user v is denoted by $S_{\mathcal{A}}(G, u, v)$. Since the union operation is commutative, $S_{\mathcal{A}}(G, u, v) = S_{\mathcal{A}}(G, v, u)$.

We define the *performance* $p_{\mathcal{A}}(G)$ of the subgraph selection algorithm $\mathcal{A}$ on the trust graph G as the ratio of the number of user pairs (u, v) where there is a directed path from u to

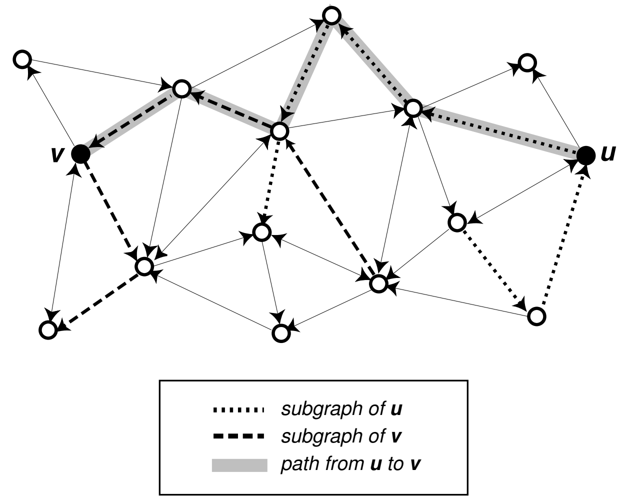

Figure 2: Merging subgraphs. When user u wants to verify the public key of user v, u and v merge their local certificate repositories, and u tries to find a certificate chain from u to v in the merged repository. In our trust graph model, local certificate repositories are represented by subgraphs and certificate chains are represented by paths. Therefore, in the model, u and v merge their subgraphs, and u searches for a path from vertex u to vertex v in the merged subgraph.

v in the merged subgraph of u and v to the number of user pairs (u, v) where there is a directed path from u to v in the trust graph. Formally:

$$p_{\mathcal{A}}(G) = \frac{\#\{(u,v) \in V \times V : u \rightsquigarrow_{S_{\mathcal{A}}(G,u,v)} v\}}{\#\{(u,v) \in V \times V : u \rightsquigarrow_G v\}}$$

where $\#$ denotes the cardinality of a set. Essentially, $p_{\mathcal{A}}(G)$ expresses which fraction of the existing directed paths in G can be reconstructed if only the subgraphs $S_{\mathcal{A}}(G, u)$ and $S_{\mathcal{A}}(G, v)$ are available when a path between u and v is requested.

Besides its performance, $\mathcal{A}$ has other important characteristics as well. One of these is the size of the subgraphs that it selects. Clearly, the performance of $\mathcal{A}$ can be increased by selecting larger subgraphs, but then, users need more memory to store their subgraphs, and this may lead to scalability problems. Another important characteristic of $\mathcal{A}$ is the amount and type of knowledge required by the users to execute it. Again, the performance of $\mathcal{A}$ can be increased by using more information about the trust graph, but the acquisition of this information may be difficult or even infeasible. Finally, the algorithm should not assign a special role to any of the nodes in the trust graph. In particular, it is not desirable that each user's subgraph contains a specific node (a virtual "center" of the trust graph). Although, in this case, any two subgraphs would intersect in this specific node, and thus, the performance of $\mathcal{A}$ would be high, the security of the system would rely on a single node, and this should be avoided.

Therefore, the design objectives of subgraph selection algorithms are the following:

- *Performance:* to achieve high performance on trust graphs that may occur in the targeted application;
- *Scalability:* to select subgraphs that have a reasonable size with respect to the size of the whole trust graph (i.e., the number of users in the system);
- *Distribution:* to rely on information that is "local" to the user that executes the algorithm; and
- *Robustness:* to avoid that the compromise of a single user or a small number of users leads to the collapse of the security of the whole system.

Clearly, there is no algorithm that is optimal with respect to all objectives; one has to find a trade-off.

5.2 The Shortcut Hunter algorithm

We expect that trust graphs that may occur in self-organized systems exhibit *small world* [26] properties (i.e., they have a small average diameter and, at the same time, they are highly clustered). *Shortcuts* play an important role in reducing the average diameter of small world graphs. A shortcut is defined as an edge, upon whose removal, the shortest undirected path between the nodes previously connected by that edge becomes strictly larger than two. By an undirected path, we mean a chain of arbitrarily directed edges. Since shortcuts are important in small world graphs, we designed a subgraph selection algorithm, called *Shortcut Hunter*, that takes into account shortcuts when building a subgraph.

The algorithm selects a subgraph that consists of two logically distinct parts: an *out-bound* and an *in-bound* path. The paths are selected in multiple rounds. When selecting the out-bound path of user u, the algorithm starts from vertex u, and in each round, it selects an outgoing edge (with its terminating vertex) that belongs to the last selected vertex. In practice, this means that u must ask the user of the last selected vertex for a list of her outgoing edges. This list can easily be provided, because each user stores her outgoing edges. The selection of the in-bound path of u is similar: the algorithm starts from vertex u, and in each round, it selects an incoming edge (with its originating vertex) that belongs to the last selected vertex. In order to make this possible, each user must also know about her incoming edges. For this reason, our algorithm requires that each user is notified whenever another user issues a certificate to her.

The selection of the next edge and its terminating or originating vertex z in each round of the algorithm is based on the number of z's shortcuts. More precisely, in each step, a vertex that has the highest number of shortcuts is chosen. A user u can determine the number of her shortcuts by obtaining information about the outgoing and incoming edges of her adjacent users (i.e., the users that belong to vertices that are connected to vertex u with either an outgoing or an incoming edge).

Below, we give a detailed description of the selection of the out-bound path. The selection of the in-bound path works in a similar way. We denote the set of vertices and the set of edges of the selected out-bound path by $V(S)$ and $E(S)$, respectively. Furthermore, we denote the set of edges of the trust graph by $E(G)$ and we assume that the algorithm is executed by user u. The length s of the selected path is a parameter. The set N contains those vertices of G that have been processed but not selected into the path.

1. Initialization: $V(S) := \{u\}$, $E(S) := \emptyset$, $N := \emptyset$, $w := u$, $i := 0$
2. $T := \{(w, z) \in E(G) : z \notin V(S) \text{ and } z \notin N\}$
3. If $T = \emptyset$, then *backtracking*:
   (a) If $w = u$, then go to step 9
   (b) Add w to N
   (c) Take the edge $(v, w) \in E(S)$
   (d) Remove (v, w) from $E(S)$, and remove w from $V(S)$
   (e) $w := v$, $i := i - 1$
   (f) Go to step 2
4. Choose the edge $(w, z) \in T$ the terminating vertex z of which has the highest number c of shortcuts (if there are several such edges, then choose one randomly)
5. If $c = 0$, then choose the edge $(w, z) \in T$ the terminating vertex z of which has the highest number of outgoing edges (if there are several such edges, then choose one randomly)
6. Add (w, z) to $E(S)$, and add z to $V(S)$
7. $w := z$, $i := i + 1$
8. If $i < s$, then go to step 2
9. Output the path $(V(S), E(S))$ and stop

5.3 Evaluation of Shortcut Hunter

We have investigated the performance of Shortcut Hunter on real PGP trust graphs that we have obtained from the web sites www.pgpi.org and www.cert.org. More precisely, we extracted the largest strongly connected component of three PGP public-key databases, and we considered these strongly connected components as trust graphs for our purposes. The performance of Shortcut Hunter on these trust graphs for various subgraph sizes is shown in Figure 3.

It can be seen that Shortcut Hunter performs very well on the two smaller trust graphs even if the size of the selected subgraphs is very small. For instance, when the size of the selected subgraphs is only 20, the performance is more than 0.97. This means that if each user stores only 20 certificates (10 out-bound and 10 in-bound) in her local repository, which are selected using Shortcut Hunter, then any user can obtain and verify the public key of any other user, using only the local certificate repositories of the two users, with approximately 0.97 probability. The performance of the algorithm on the largest trust graph is significantly worse;

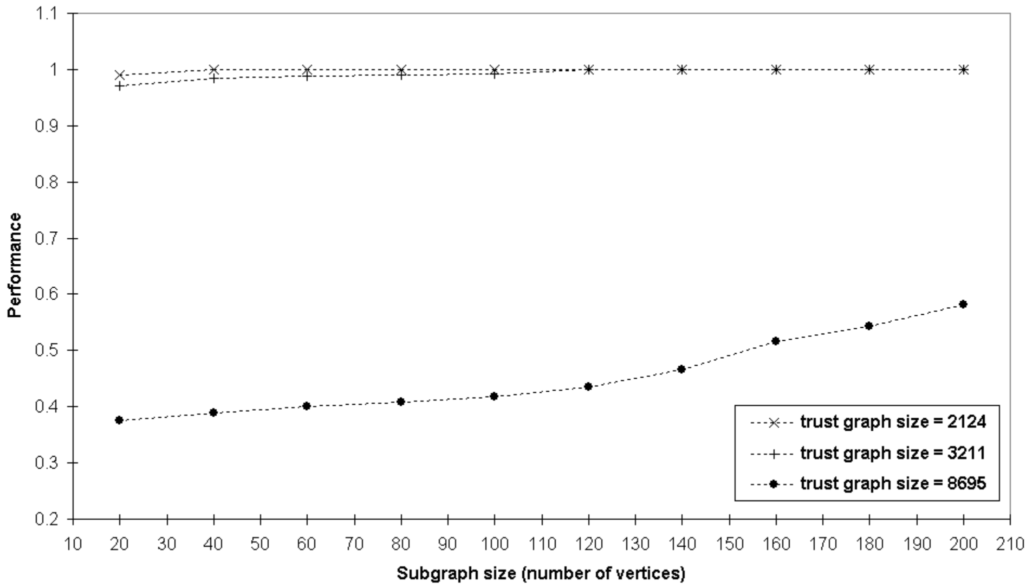

Figure 3: The performance of Shortcut Hunter in real PGP trust graphs

however, it shows an increasing tendency as the subgraph size increases.

The pure Shortcut Hunter algorithm builds a subgraph that consists of a single out-bound and a single in-bound path. We modified it in such a way that it builds a subgraph that consists of several vertex disjoint out-bound and vertex disjoint in-bound paths. We call this modified algorithm Star Shortcut Hunter (since the subgraph that it builds resembles a star). More precisely, when running from a vertex u, Star Shortcut Hunter builds $p_{out} = \min(n_{out}, c)$ vertex disjoint out-bound and $p_{in} = \min(n_{in}, c)$ vertex disjoint in-bound paths where n_{out} and n_{in} denote the number of u's outgoing and incoming edges, respectively, and c is a constant. The length ℓ of each path is equal to $\lceil s/(p_{out} + p_{in})\rceil$, where s is the required size of the resulting subgraph (which is an input of the algorithm). Note that if $c = 1$, then Star Shortcut Hunter is equivalent to the pure Shortcut Hunter algorithm.

The operation of Star Shortcut Hunter is based on the algorithm described in the previous subsection. That algorithm is executed p_{out} times to build p_{out} out-bound paths of length ℓ. In order to ensure that the selected out-bound paths are disjoint, at the beginning of each execution, the set N is initialized with the set of vertices selected so far, instead of resetting it to $\emptyset$ each time. The selection of the in-bound paths is based on similar principles. The performance of Star Shortcut Hunter when $c = 10$ is shown in Figure 4.

It can be seen that Star Shortcut Hunter performs better on the largest trust graph than Shortcut Hunter does. Its performance on all three trust graphs is more than 0.95 when the size of the selected subgraph is around two times the square root of the size of the trust graph. This shows that Star Shortcut Hunter can achieve a high performance, yet it is scalable. In addition, the user who executes it needs to have only local knowledge of the trust graph. Namely, the user needs information only about the number of shortcuts of the neighbors of the last selected vertex, in order to select the next vertex into the subgraph. However, regarding robustness of Star Shortcut Hunter (and Shortcut Hunter), it seems that some vertices are selected more often into the subgraphs than others.

5.4 Dishonest users

So far, we have assumed that each user is honest and does not issue false certificates. Relaxing this assumption requires the introduction of some sort of authentication metric into our model. An authentication metric is a function μ that takes two users u and v and a trust graph G as inputs, and returns a numeric value $\mu(u, v, G)$ that represents the assurance with which u can obtain the authentic public key of v using the information in G. For instance, μ could return the number of vertex disjoint paths from u to v in G. For an overview of existing authentication metrics and an insight into their design principles, we refer to [23].

The definition of the performance of a subgraph selection algorithm $\mathcal{A}$ on a trust graph $G = (V, E)$ when authentication metric μ is used can be defined as follows:

$$p_{\mathcal{A},\mu}(G) = \frac{1}{\#W} \sum_{(u,v)\in W} \frac{\mu(u, v, S_{\mathcal{A}}(G, u, v))}{\mu(u, v, G)}$$

where $W = \{(u, v) \in V \times V : \mu(u, v, G) \neq 0\}$. $p_{\mathcal{A},\mu}(G)$ represents the average of the ratios of the assurance obtained when only information from the local repositories is used

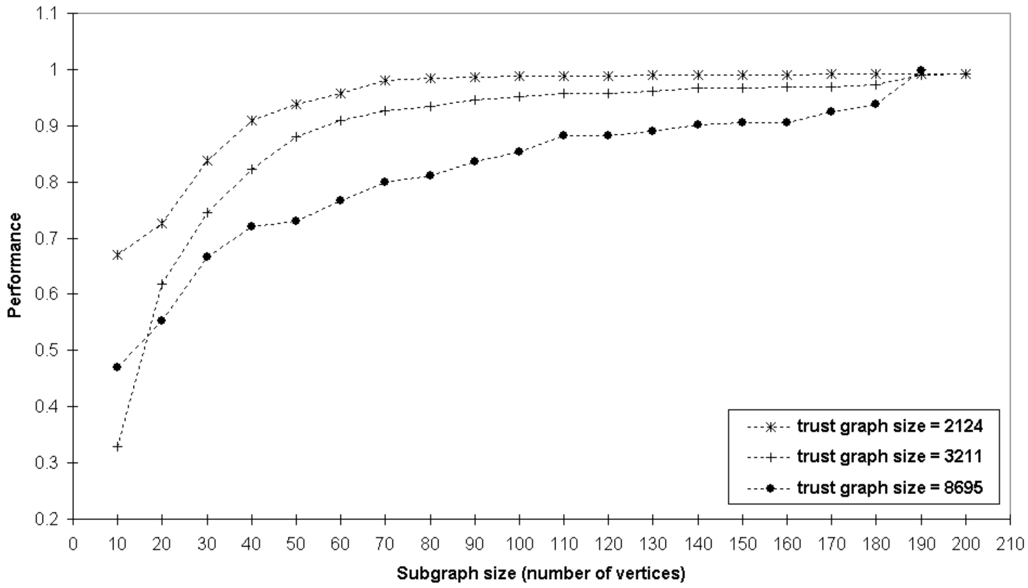

Figure 4: The performance of Star Shortcut Hunter when $c = 10$ in real PGP trust graphs

to the assurance obtained when information from the whole trust graph is used. If $p_{\mathcal{A},\mu}(G)$ is close to one, then essentially the same assurance can be obtained using the local repositories as using the whole trust graph.

Note that the above definition of the algorithm performance is a generalization of our previous definition given in Subsection 5.1. In fact, in that subsection, we implicitly used the binary metric

$$\mu_0(u, v, G) = \begin{cases} 1 & \text{if } u \rightsquigarrow_G v \\ 0 & \text{otherwise} \end{cases}$$

and it is easy to verify that $p_{\mathcal{A},\mu_0}(G) = p_{\mathcal{A}}(G)$ for any $\mathcal{A}$ and G.

In the future, we intend to use more complex authentication metrics that better model the effect of dishonest users on the public-key distribution system. Whether the proposed algorithms prove to be efficient with respect to these metrics is a question that we intend to explore.

6. CONCLUSION

We have surveyed the threats and possible solutions for the basic mechanisms and for the security mechanisms in mobile ad hoc networks. As for the latter, we developed the idea of a self-organized public-key infrastructure. Our system is similar to PGP in the sense that public-key certificates are issued by the users. However, as opposed to PGP, we do not rely on certificate directories for the distribution of certificates. Instead, in our system, certificates are stored and distributed by the users. We presented two algorithms that users can use to build their local certificate repositories. We showed that the algorithms achieve a high performance on real PGP trust graphs, which means that any pair of users can find certificate chains to each other using only their local certificate repositories with a high probability. In addition, the size of the local certificate repositories is small compared to the total number of users in the system.

It has to be noted that this approach is not confined to ad hoc networks. Indeed, it can be applied in all cases when security has to be self-organized. Peer-to-peer applications are a wonderful example thereof.

The analysis carried out in this paper raises a more general and philosophical question. As we have seen, it is tempting to think of mobile ad hoc networks as being self-organized. The most relevant example of self-organization in the area of security is PGP; however, PGP has remained confined so far primarily to the community of computer literate users.

In the global domain of networking, the most impressive example of a widespread self-organized system is the World Wide Web; indeed, with the Web, any user can read and publish information, and contribute to the evolution of the contents in a totally decentralized way. Standardization bodies do accomplish an important work of refinement of the building blocks (such as the format of URLs, HTML and HTTP), but no organization *runs* the Web; the underlying IP network is operated, but the Web as such is not. Can security be brought to self-organized mobile ad hoc networks in as decentralized a way as the hypertext was brought to the Internet? If yes, such a powerful property could also be misused; what are then the mechanisms to be put in place to prevent this to happen? Both questions are currently on our research agenda.

7. ACKNOWLEDGEMENTS

We express our gratitude to the students Chinmoy Dutta, Ratul Guha, Giuseppe Schiavello and Renato Vilan for their contribution to the analysis of the PGP trust graphs and the study of the algorithms to build the subgraphs. The first author would like to thank Jean-Yves Le Boudec for valuable discussions on the societal implications of self-organization.

8. REFERENCES

[1] The Terminodes Project. www.terminodes.org.

[2] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, 1996.

[3] N. Asokan and P. Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23:1627–1637, 2000.

[4] E. Ayanoglu et al. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Transactions on Communications*, 41(11), 1993.

[5] L. Blažević, L. Buttyán, S. Čapkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self-organization in mobile ad hoc networks: The approach of Terminodes. *IEEE Communications Magazine*, June 2001.

[6] C. Boyd and A. Mathuria. Key establishment protocols for secure mobile communications: a critical survey. *Computer Communications*, 23:575–587, 2000.

[7] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad hoc networks. In *Proceedings of MobiHOC*, 2000.

[8] L. Buttyán and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne, 2001.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, 1976.

[10] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, pages 107–125, 1992.

[11] A. Ephremides, J. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 1987.

[12] L. M. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous networking: an application-oriented approach to ad hoc networking. *IEEE Communications Magazine*, June 2001.

[13] M. Girault. Self-certified public keys. In *Proceedings of Eurocrypt'91*, 1991.

[14] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli. Toward self-organized mobile ad hoc networks: The Terminodes Project. *IEEE Communications Magazine*, January 2001.

[15] M. Jacobsson and S. Wetzel. Security weaknesses in Bluetooth. Technical report, Bell Labs, January 2001.

[16] S. Jiang, N. Vaidya, and W. Zhao. Routing in packet radio networks to prevent traffic analysis. In *Proceedings of the IEEE Information Assurance and Security Workshop*, West Point, NY, July 2000.

[17] J. Linn. Privacy enhancement for Internet electronic mail: Part I–IV. Internet RFC 1421–1424, 1993.

[18] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM*, 2000.

[19] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distributions systems. *Transactions of the IECE (Japan)*, pages 107–125, 1986.

[20] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[21] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of MOBICOM*, 2001.

[22] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Trusting mobile user devices and security modules. *IEEE Computer*, February 1997.

[23] M. Reiter and S. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2):138–158, 1999.

[24] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO'84*, 1984.

[25] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, 1999.

[26] D. Watts. *Small Worlds*. Princeton University Press, 1999.

[27] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of MOBICOM*, 2000.

[28] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November/December 1999.

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