A Distributed Light-Weight Authentication Model for Ad-hoc Networks

André Weimerskirch[2,1] and Gilles Thonet[1]

[1] Accenture Technology Labs, Sophia Antipolis, France
gilles.thonet@accenture.com
[2] Communication Security Group, Ruhr-Universität Bochum, Germany
weika@crypto.ruhr-uni-bochum.de

The 4th International Conference on Information Security and Cryptology (ICISC 2001), 6-7 December 2001, Seoul, South Korea

Abstract. In this work we present a security model for low-value transactions in ad-hoc networks in which we focus on authentication since this is the core requirement for commercial transactions. After a brief introduction to the ad-hoc networking paradigm we give a survey of various existing models and analyze them in terms of scope and applications. It is concluded that no appropriate and satisfactory model for low-value transactions in ad-hoc networks has been developed to date. Our new model is derived from previous models dealing with trust and security in ad-hoc networks, and does not require devices with strong processors as public-key systems. We base the model on a recommendation and reference protocol that is inspired by human behavior and that is in accordance with the very nature of ad-hoc networks.

Keywords: Ad-hoc network, security, low-value transactions, trust, authentication, public-key

1 Introduction

Today microprocessors can be found almost everywhere. It is a well established trend to embed a microprocessor in nearly all electronic devices such as cellular phones, televisions and video recorders. In the future this might extend to everything from coffee makers to washing machines which are then connected by a network. In combination with already existing computing devices such as desktop computers, notebooks and Personal Digital Assistants (PDAs) this could form an extremely widespread wireless network of mobile and static devices communicating without fixed infrastructure or centralized administration. In such a self-organized network each node relies on its neighbor nodes to keep the network connected, e.g., each node routes data packets from its neighbors. Furthermore each node might take advantage of the services offered by other nodes. This type of network is usually called an *ad-hoc network* and is particularly useful when

a reliable fixed or mobile infrastructure is not available – e.g. after a natural disaster – or too expensive.

The security issues for ad-hoc networks are different than the ones for fixed networks. While the security requirements are the same, namely availability, confidentiality, integrity, authentication, and non-repudiation, their provision must be approached differently for ad-hoc networks. This is due to system constraints in mobile devices and frequent topology changes in the network. System constraints include low-power microprocessor, small memory and bandwidth, and limited battery power. In addition to usual denial of service attacks where a node is flooded by a vast amount of requests, availability in ad-hoc networks can be threatened by radio jamming and battery exhaustion [16]. Closely related to security is secure routing. Since each node of the network potentially acts as a repeater, a malicious node might heavily affect packet routing.

While there are many security issues in ad-hoc networks this paper focuses on *entity authentication* [14]. This is the core requirement for integrity, confidentiality and non-repudiation — if you do not know who you are talking to it does not make sense to establish a secure channel. The paper is organized as follows. Section 2 is a brief introduction to ad-hoc networking and its importance as future mobile networking model. Section 3 gives an overview of existing security models for ad-hoc networks. Then a discussion of the application domain of these models follows in Section 4. Section 5 introduces our new approach and the final section concludes the paper.

2 Ad-hoc Networks

As said previously an ad-hoc network is a wireless network made up of mobile hosts that do not require any fixed infrastructure to communicate. The basic idea behind this model is not recent: as early as the 70s, ad-hoc networks were called *packet radio networks* and were investigated for military applications almost exclusively – PRnet, developed by the American Defense Advanced Research Projects Agency (DARPA), is probably the most famous example [13]. Then, when designing the 802.11 standard for Wireless Local Area Networks (WLAN) the IEEE replaced the term packet radio network by ad-hoc network, hoping to forget the military connotation of the former one. Ad-hoc networks are frequently associated with self-organization, which means that they run solely by the operation of end-users (like the old citizen-band voice analogue network). Although pure self-organization is not required to form an ad-hoc network this feature should be understood as a basic requirement for decentralization: hosts must be capable to enter and leave the network without referring to a central authority. It is important to note that ad-hoc networks would likely neither be a replacement nor an alternative to current and future infrastructure-based networks. Their scope is more to complement the latter in cases where cost, environment or application constraints require self-organized and infrastructure-less solutions.

2.1 Main Issues

Ad-hoc networks include various architectures that range from fully mobile and decentralized radio, access, and routing technologies – the ones being investigated as part of the standardization work carried out by the Internet Engineering Task Force (IETF) [15] – to more infrastructure-dependent standards that include an ad-hoc mode – e.g. IEEE 802.11 [11] and HiperLAN2 [10].

Here are some of the basic features one can expect to find in most ad-hoc networks:

- Communication links are wireless to guarantee mobility. Accordingly, their dependability and capacity have to be carefully scrutinized.
- Ad-hoc networks act independently from any provider. However, access points to a fixed backbone network are expected to be available if required.
- Because they do not rely on a fixed infrastructure mobile hosts have to be somehow *cooperative*. This ranges from very simple schemes for short-range networks to highly cooperative strategies in case of multi-hop wide-area networks.
- The network topology may be very dynamic, making the links and routes very unstable.
- Power management is an important system design criterion. Hosts have to be power-aware when performing such tasks as routing and mobility management.
- Finally, security is a critical issue because of the weak connectivity and of the limited physical protection of the mobile hosts. This is the focal point of this paper.

2.2 Potential Applications

Because packet radio networks have been developed initially for military purposes, potential applications are very often associated with critical situations such as battlefields and damaged areas. Other prospective applications are still at an early stage, as summarized in the following list:

- Military applications: communications between soldiers, soldiers monitoring, sensor networks for target detection and identification, ...
- Emergency situations where the existing network infrastructure is not reliable or has been damaged due to geopolitical instability, natural or man-made disaster, ...
- Provision of wireless connectivity in locations where cellular networks present an insufficient coverage, are more expensive or are wanted to be by-passed (e.g. for privacy reasons).
- Distributed networks for data collection and device monitoring: sensor networks, home networks, inter-vehicle communications, supply chain management, ...
- Creation of instant and temporary networks for ad-hoc meeting, conferences or brainstorming.

3 Previous Work

This section presents various security models for ad-hoc networks. Since a strong relationship can be identified between trust and security inherent to ad-hoc networks we start by introducing basic properties of trust and their role for security. Then, we describe a distributed trust model that we will use later in this work.

3.1 Distributed Trust Model

Before describing the Distributed Trust Model [1] it is worth mentioning some research results about trust [12]. Trust in human beings differs from trust in an IT system. We trust a human if we believe him to be benevolent, and we trust an IT system if we believe it is robust against malicious human attackers. When we trust an entity we distinguish the target of trust and its classification. The target of trust is the entity we trust, while the classification describes exactly for which aspect the entity is being trusted – while I might trust Bob in repairing a computer I might not trust him in repairing a car. Furthermore, there might be a value of trust which describes how much we trust an entity, or a ranking expressing that we trust entity A more than entity B with respect to some criteria.

Trust in distributed systems should as much as possible be based on knowledge to prevent irrational trust based on faith. There can be a hierarchy of trust relationships: explicit trust relationships are direct while implicit relationships rely on an indirect trust path. Trust can be derived by establishing a direct trust relationship using an indirect path.

The *Distributed Trust* Model [1] is a decentralized approach to trust management and uses a recommendation protocol to exchange trust-related information. The model assumes that trust relationships are unidirectional and taking place between two entities. The entities make judgments about the quality of a recommendation based on their policies, i.e., they have values for trust relationships. Also, trust is not absolute, e.g., an entity can change the trust value it received as a recommendation. The policy is not communicated, so it might not be understandable for other entities which prevents it from being misused. The recommendation protocol works by requesting a trust value in a trust target with respect to a particular classification. After getting an answer an evaluation function is used to obtain an overall trust value in the target. The protocol also allows recommendation refreshing and revocation. To do so the recommender sends the same recommendation with another recommendation value, or a neutral value to revoke. The model is suited to establishing trust relationships that are less formal, temporary, or targeting ad-hoc commercial transactions.

3.2 Password-Based Key Agreement

The work developed in [3] addresses the scenario of a group of people who want to set up a secure session in a meeting room without any support infrastructure. Desirable properties of a protocol that solves this problem are:

- *Secrecy*: only those entities that know an initial password are able to learn the session key. Perfect forward secrecy requires that an attacker who compromises one member of the group and learns all his permanent secret information is still unable to recover the session key.
- *Contributory key agreement*: the session key is formed of contributions from all entities. This ensures that if only one entity chooses its contribution key randomly all other entities will not be able to make the key space smaller.
- *Tolerance to disruption attempts*: the protocol must not be vulnerable to an attacker who is able to insert messages. It is assumed that the possibility of modifying or deleting messages in such an ad-hoc network is very unlikely.

The work describes and introduces several password-based key-exchange methods that meet these requirements. The core idea of the protocol is as follows. A weak password is sent to the group members. Each member then contributes part of the key and signs this data by using the weak password. Finally a secure session key to establish a secure channel is derived without any central trust authority or support infrastructure.

3.3 Resurrecting Duckling Security Policy

This policy was introduced in [16] and extended in [17]. It is particularly suited for devices without display, and for embedded devices which are too weak for public-key operations. The fundamental authentication problem is a solved by a secure transient association between two devices establishing a master-slave relationship. It is secure in the sense that master and slave share a common secret, and transient because the association can be terminated by the master only. Also a master can always identify its slave in a set of similar devices.

The proposed solution is called the *Resurrecting Duckling* model. The duckling is here the slave device while the mother duck is the master controller. The duckling will recognize as its mother the first entity that sends it a secret key on a secure channel, e.g., by physical contact during the device initialization. This procedure is called *imprinting*. The duckling will always obey its mother, who tells it whom to talk to through an access control list. The bond between mother and duckling is broken by death after which the duckling accepts another imprinting. Death may be caused by the mother itself, a timeout or any specific event. The whole security chain corresponds to a tree topology formed of hierarchical master-slave relationships. The root of the tree is a human being controlling all devices, and every node controls all devices in its subtree. However, if one relationship is broken the relationship to the whole subtree is also broken.

This security model can be applied to very large ad-hoc networks, e.g. networks consisting of smart dust devices [18]. A possible scenario is a battlefield of smart dust soldiers (acting as slaves or siblings) and their general (acting as the master). The master allows its slaves to communicate by uploading in each of them a highly flexible policy so that sibling entities become masters and slaves for a very short time, enough to perform one transaction. The mother duck gives the ducklings credentials that allow them to authenticate themselves.

3.4 Distributed Public-Key Management

Key management for established public-key systems requires a centralized trusted entity called *Certificate Authority* (CA). The CA issues certificates by binding a public key to a node's identity. One constraint is that the CA should always be available because certificates might be renewed or revoked. Replicating the CA improves availability. However, a central service runs contrary to the distributed structure of ad-hoc networks.

Reference [20] proposes to distribute trust to a set of nodes by letting them share the key management service, in particular the ability to sign certificates. This is done using *threshold cryptography* [6]. An $(n, t+1)$ threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation so that any $t+1$ parties can perform this operation jointly whereas it is infeasible for at most t parties to do so. Using this scheme the private key k of the CA is divided into n shares $(s_1, s_2, \ldots, s_n)$, each share being assigned to each special node. Using this share a set of $t+1$ special nodes is able to generate a valid certificate. As long as t or less special nodes are compromised and do not participate in generating certificates the service can operate. Even if compromised nodes deliver incorrect data the service is able to sign certificates. Threshold cryptography can also be applied to well known signature schemes like the *Digital Signature Standard* (DSS) [7].

Another approach introduced in [9] presents a self-organized public-key infrastructure. The system replaces the centralized CA by certificate chains. Users issue certificates if they are confident about the identity, i.e. if they believe that a given public key belongs to a given user. Each user stores a list of certificates in its own repository. To obtain the certificate of another entity the requester builds a certificate chain using his repository list and implicitly trusted entity's lists until a path to an entity that has the desired certificate in its repository is found.

4 Scope of Security Models

4.1 Distributed Trust Model

The Distributed Trust Model describes how to establish trust relationships. It can be used on top of security models such as public-key systems. It is applicable to any distributed system and does not target specifically ad-hoc networks. An implementation for ad-hoc networks would be particularly vulnerable to malicious and compromised agents. Since an ad-hoc network has a very flexible topology it is also unclear if enough trustworthy entities are available to obtain a recommendation. Although it is possible to make a broadcast asking for targets offering a special service this can only be successful if the available network of explicitly and implicitly trusted entities is large enough. Finally the requester could ask for information regarding what other entities think about each other. This could be data that other entities would not want to reveal.

4.2 Password-Based Key Agreement

This model perfectly works for small groups. Authentication is done outside the IT system, e.g., the group members authenticate themselves by showing their passports or based on common knowledge[1]. The model does not suffice anymore for more complicated environments, though. Groups of people who do not know each other, or pairs of people who want to have confidential exchanges without the rest of the group be able to eavesdrop on the channel, are two examples. Another problem arises for large groups or groups at different locations. The secure channel to distribute the initial password is not available anymore. At this point it seems that existing support infrastructure is required to set up a secure channel.

4.3 Resurrecting Duckling

The Resurrecting Duckling scheme is an appropriate model for a well defined hierarchy of trust relationships. It particularly suits inexpensive devices that do not need a display or a processor to perform public-key operations. This perfectly works for a set of home devices, for instance. However, more flexible ad-hoc networks may not contain explicit trust relationships between each pair of nodes or to a centralized entity like the mother duck. Deploying a comprehensive network consisting of a hierarchy of a global mother duck and multiple subsidiary local mother ducks is very similar to a public-key infrastructure, where the mother ducks correspond to CAs, with all its advantages and drawbacks. Even the battlefield scenario raises some problems. Here the soldiers are siblings and obey to their mother, the general. If one soldier device wants to authenticate to another device it has to present its credentials [17]. The second device can then check the credentials by using its policy. But what happens if not all soldiers use the same credentials, i.e. the same secret key, to prevent it to be stolen by the enemy? If all devices use the same key the other side might invest considerable effort doing some physical attack [2] to recover the key because it would compromise all nodes. Since the devices cannot hold a list of all valid credentials it seems that a further authentication method is needed.

4.4 Distributed Public-Key Management

The Threshold Key Management system is a way to distribute a public-key system. For high-value transactions public-key systems are certainly the only way to provide a satisfactory and legal security framework. As mentioned earlier in this paper trust should as much as possible be based on knowledge. Since two entities that never met before cannot have common knowledge – a shared secret – they both have to trust a central entity, e.g., a CA to substitute this knowledge. When a user wants to prove his identity to a CA he goes there with his public key and shows his passport. The CA proves his identity and then

[1] In this case friendship or just knowing each other is considered as common knowledge.

binds his identity to his public key and signs the certificate. But how is this done when the CA is distributed? The user must prove his identity to all special nodes to prevent that a compromised node passes on faulty information. But if the CA signs certificates without proving the identity the model cannot be used for high-value transactions.

The self-organized public-key infrastructure [9] shows similar problems. To make the system bullet-proof, the entity's identity has to be checked in the real world before users issue certificates. Furthermore, it is assumed that the certificate requester trusts each node in the recommendation chain. Finally a significant computing power and time is consumed to obtain a certificate going through the certificate chain. Each node in the chain has to perform public-key operations, first to check the received certificate for authentication (signature verification) and then to sign it before forwarding it (signature generation). This cannot be done in parallel but only one after the other until the certificate went along the entire chain.

Despite its centralized nature a central CA is preferable for applications with high-security demand. To ensure high availability the CA can be replicated. The replicated CAs are as secure as the original CA as long as the replication process is not vulnerable to attacks. The private key of the CA does not get weaker after replication. Much research has been done about efficient public-key systems – for example, a public-key system for mobile systems is presented in [8].

4.5 Different Scopes for Different Applications

No model ensures authentication in an ad-hoc network in every environment. Depending on the situation users can select the appropriate system. While for a group meeting in a small conference room the password-based key-exchange will work perfectly, for a network defined by a hierarchy of trust relationships the resurrecting duckling policy is the best alternative. To guarantee secure transactions the appropriate choice is to use a public-key system involving the hassle of getting certificates, to use a device that can perform such operations and find a reliable connection to the CA to check for revoked or renewed certificates.

Also, using no security at all is an acceptable approach for many applications. A good illustration is a huge network of cell phones where free calls can be made without using a phone service provider. Once the network has enough nodes and enough redundant connections, the no-security approach might be the best one since every user benefits from routing other users' data packets. However, none of the models presented up to now is a good solution for low-value transactions in ad-hoc networks. The next section fills this gap.

5 Distributed Light-Weight Authentication Model

5.1 Main Goals

Our new model is strongly based on human behavior when dealing with trust since the human society itself can be seen as an ad-hoc network. We use the

recommendation protocol from the Distributed Trust Model to establish trust relationships and extend it by a request for references. Each entity maintains a repository of trustworthy entities such that a path between two arbitrary entities can be found by indirectly using the repositories of other entities. This idea has also been recently used for a self-organizing public-key system [9]. We also bring in the idea of threshold cryptography in the sense that compromised nodes, as long as their number is below a threshold, cannot harm the result of the operation. Since we are trying to establish a general model for low-value transactions our main goal is not to make transactions perfectly secure but rather to make the attacker's cost to get falsely authenticated higher than the value of the transaction. Thus, an attacker should not be able to make the effort once to break multiple transactions, but he should have high cost to break each single transaction.

The starting point is human behavior. To check out another person, people usually ask friends about that person. Also they will ask the target person for references. To check the identity of another person people will ask about common knowledge, e.g., a secret password or details of previous transactions. If the other person knows details which only the right person can know he is considered to be this person. If there is no explicit common knowledge a trust relationship can be derived using a trusted third party. This can be the CA in case of a public-key system or the mother duck in case of the hierarchical trust relationships. In our case trust relationships will be derived using a trust chain in the trust network of friends and friends' friends. A cooperation and feedback system introduces quality and user responsibility for recommendations to the model.

5.2 Model Description

The model works as follows. Let's assume Alice wants to check if Bob is who he pretends to be, i.e., she wants to verify Bob's identity. Note that in this case Alice and Bob are actually devices representing a user, and not human beings. Therefore users should authenticate themselves to the devices. It is worth mentioning that the model is applicable to users instead of devices but it would require some effort from the user. Alice starts asking Bob about common knowledge. This can be a secret key, but also knowledge about a recent transaction. If there is common knowledge Bob can prove his identity.

Recommendations Otherwise, Alice starts requesting recommendations from nodes taken from her list of trustworthy entities. The classification of these requests is if the asked person believes that Bob is who he pretends to be, i.e., a request for a recommendation about Bob's identity. Note that in the simplest case possible return values can be "yes" or "no". Let's assume one of the devices Alice asks is Cathy. She can check Cathy's identity by asking her about common knowledge, e.g., a shared secret key or part of the content of their last transaction. This could be done using a challenge-response protocol [14]. Assuming that this transaction was encrypted its content cannot be eavesdropped by

an attacker. Cathy then looks if Bob is on her list of trustworthy people and checks his identity, or forwards Alice's request in the same manner. Once an entity is found that knows Bob and can prove his identity, let's call him Dan, the information is sent back to Alice. She also obtains the name of all entities in the recommendation chain. To introduce anonymity the nodes in the recommendation chain could only forward the length of the chain and the result "yes" or "no". However, this would affect the feedback system that is described later on. To include some randomness Alice might ask random entities about Bob's identity. She might also ask Cathy and all other entities in the recommendation chain to do the same.

References Asking for references is the next human behavior characteristic we use in our model. Alice can ask Bob to give his references, i.e., other devices he has done transactions with recently. Alice can then ask these devices if they know Bob. Since Bob (in this case the user behind the device) could easily set up the entities he gives as references, Alice can ask again her network of trustworthy entities if they know the references. Of course a good reference would be an entity that has a relationship with Alice and Bob. To be more convinced Alice could ask the references for references again and so on. Once a link between Alice's trust network and Bob's reference network is found (both in a recursive sense) a direct relationship between Alice and Bob can be derived.

Since Alice broadcasted her request to many nodes, and also Cathy broadcasted the request, the number of broadcasted requests is exponential. Therefore the chain length to find a relationship between Alice and Bob should be low. It has been shown that networks such as ad-hoc networks have a very small degree of separation [19]. This means that the chain between two arbitrary entities in an ad-hoc network has low length. Using both requests for recommendations and references the two trust networks are browsed starting at Alice and Bob until a connection is found. This keeps the number of involved nodes low. To further reduce the network traffic Alice and Cathy might have rankings for trustworthy entities. Alice might ask Bob for a recommendation if Bob is a cell phone but not if he is part of an on-line store. Also there can be nodes in the network particularly suited for recommendations, e.g., immobile servers that perform many transactions. These servers may be highly available through replication. Central servers, however, are not necessary for the protocol. After Alice has received the results of her request she has to evaluate the data.

Trust Evaluation The evaluation phase allows Alice to decide if she believes in Bob's identity or not – i.e. the evaluation function maps information of all received data of the recommendation chains to a "yes" or a "no". For more sophisticated systems the function might output an upper limit for a transaction value when dealing with Bob. Before evaluation Alice can check the recommendation chains for suspicious nodes and if necessary ignore these chains for evaluation. The identification chains of Bob's references are included in the evaluation, but can be differently weighted than recommendations about Bob's

identity. For the evaluation function we want to bring in the idea of threshold cryptography. If Alice receives n answers to her request, then we want that the result be not influenced by t or less malicious agents, where n is considerably larger than $2t$. How this is achieved is due to the local policy of Alice, which does not need to be public. A simple approach is to cut off t "yes" and t "no" results to the question about Bob's identity and evaluate the remaining values. A very suspicious person could configure its device such that it requires only "yes" answers while less suspicious users would allow some errors. Since an (n, t) scheme requires that exactly n nodes respond we will interpret such a scheme in the sense that $t/n \cdot 100\%$ or less compromised nodes that respond cannot affect the result. Unfortunately it is impossible to distinguish a malicious node that does not respond from a node that does not respond because of other reasons, e.g., to cover existing relationships. Management of the the trust relationships and credentials can be handled by a system as proposed in [4].

Secure Channel and Routing Once an entity is authenticated – or both entities have authenticated mutually – a secure channel can be initiated. Since we assumed that there is no common knowledge between both entities a perfectly secure channel cannot be set up. In many cases it is sufficient not to use encryption at all. Otherwise, Alice can send Bob a secret over the trustworthy path used to derive a relationship. Bob then sends back another secret using a random path. By combining these secrets Alice and Bob can derive a shared secret key, as shown in [3]. Only trustworthy entities can obtain the secret key by eavesdropping Bob's answer. If strong processors are available a public-key method can be used. Alice generates a public/private key pair and sends Bob the public key over the trustworthy path. Bob generates a random secret, encrypts it using Alice's public key, and sends this to Alice over the trustworthy channel. Again, this method is only vulnerable to a man-in-the-middle attack mounted by a trustworthy device. The secure channel ensures confidentiality and integrity but no legal non-repudiation.

The proposed scheme can also be used for secure routing. Using a public-key method to make routing robust might be too expensive and excludes devices with weak processors. Therefore we suggest not to use any authentication method unless the loss of packets rises above a threshold. At this point the nodes might want to check out their neighbor. They can make requests about their identity, or use a similar recommendation scheme to ask about their abilities in routing data.

Cooperation Now one might ask why any entity should participate in a request and use its battery power. We suggest a combination of incentive and punishment. Devices that give recommendations can also expect to receive answers to their requests. A device that does not want to waste battery power and therefore never answers to any request will be removed from the list of trustworthy nodes – or even put on a list of untrustworthy devices – and then receive very few or

no answers to its requests. The ability and quality to authenticate other entities is therefore weakened.

Furthermore entities might receive rewards for good recommendations. For example, an on-line shop might give Cathy and Dan a prize reduction for each successful recommendation. Existing incentive schemes – for instance the ones detailed in [5] – that stimulate cooperation in ad-hoc networks can certainly be applied to our model. However, there is no solution yet for nodes that do not cooperate because they want to hide existing transaction relationships.

Feedback Information Finally, to make the system more robust and protect it from compromised nodes, we introduce feedback messages. For example, if Alice receives from Dan via Cathy a positive recommendation about Bob's identity, but then gets cheated by Bob, she can inform Cathy and Dan about their wrong recommendation, and she might also put Bob, Cathy and/or Dan on a list of suspicious devices. Cathy and Dan can then take actions to find the security leak. The reward and feedback system introduces some quality and responsibility function to the model. The model also allows refreshing and revocation. If Dan validates Bob's identity but later on gets cheated by Bob, Dan can send a special message to Alice with negative result about Bob's identity. Of course he can also broadcast this message. Note that this does little damage to the real Bob if he can prove his identity to Alice using common knowledge.

Authentication of Users We have not considered authentication of users by now. To solve this issue we require that users authenticate themselves to the device. This can be done using a strong one-way function, and furthermore by encrypting the file system. A problem might arise if a device is sold. Since the new user must be able to authenticate himself to the device it seems that he also takes over all trust relationships of the previous owner. To prevent this from happening the previous owner can change the device ID and delete all data on the device. The new owner can change back the ID but he cannot recover the data that include shared knowledge with the trusted devices. Therefore authentication with trusted devices will fail. This also makes clear that a simple device ID spoofing is not sufficient for a successful attack.

5.3 Security Analysis

An evaluation of the security level is very hard since the model is not based on pure mathematical foundations and also leaves the individual security policy open. So what does an attacker have to do to get falsely authenticated? Let's assume Alice wants to authenticate a device that claims to be Bob but that is actually Cathy. Cathy first had to fake references. If Alice asks these references for another reference the number of references she had to fake rises exponentially. Note that it is not enough to just fake one device and give it many IDs. Since a reference is asked about knowledge – i.e. about its history – faking a reference is much harder than just faking its ID. Furthermore Cathy had to

compromise nodes from Alice's list of trustworthy entities. Finally she also had to compromise or set up enough random nodes such that the probability that Alice exactly asks these nodes is high enough. The probability is configurable by Alice by setting the threshold value (n, t) in her local security policy. Alice can also configure individually other parameters of her security policy. For example, she can reject dealing with Bob if the number of "no" requests about Bob's identity is above the threshold of t. An attacker can do a local attack by setting up a bunch of nodes at one geographical point to increase his chances for many successful attacks in this area. In this case the feedback system is able to indicate the attack before the attacker is able to regain the effort and money it took him to set up the attack. The initial cost of the attack is likely too high for the average attacker. However, we understand that our model inherits inherent problems of the Distributed Trust Model, in particular the concerns that entities need to reveal their transaction relationships. Furthermore, establishment of a secure channel is not perfectly secure. For secure channel establishment we replace a shared knowledge or trusted third party by a path in the network of trusted entities. The longer the path the higher the probability of a malicious entity among them. Therefore much effort has to be spent on finding efficient algorithms and well-sized repository lists, e.g., as done in [9]. Another challenge is the essential feedback system. It has to be efficient, detect fraud quickly, and take appropriate action to prevent repeated attacks.

6 Conclusions

In this work we have dealt with security in ad-hoc networks. We have focused on authentication since this is the core requirement to initiate a secure channel. We have described several existing models and analyzed their scope, possible scenarios but also inherent problems. Finally we have proposed a new authentication model for low-value transactions. This model is inspired by human behavior and it is consistent with the ad-hoc network paradigm. It makes use of recommendations and references to derive a trust relationship, and is scalable with respect to security. A cooperation and feedback system introduces quality and responsibility. The requirements in the device's hardware are very low. Using our model, the nodes in an ad-hoc network can establish a secure channel.

References

1. A. Abdul-Rahman and S. Hailes. A Distributed Trust Model. *New Security Paradigms Workshop 1997*, ACM, 1997.
2. R. Anderson and M. Kuhn. Tamper resistance – a cautionary note. *2nd USENIX Workshop on Electronic Commerce*, 1996.
3. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. *Computer Communications 23*, 2000.
4. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. *The 17th Symposium on Security and Privacy*, IEEE Computer Society Press, 1996.

5. L. Buttyán and J.-P. Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. *Technical Report DSC/2001/001*, Swiss Federal Institute of Technology – Lausanne, Department of Communication Systems, 2001.
6. Y. Desmedt and Y. Frankel. Threshold cryptosystems. *Advances in Cryptology – Crypto '89*, LNCS 435, Springer-Verlag, 1990.
7. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Advances in Cryptology – Eurocrypt '96*, LNCS 1070, Springer-Verlag, 1996.
8. G. Horn and B. Preneel. Authentication and Payment in Future Mobile Systems. *Computer Security - ESORICS '98*, LNCS 1485, Springer-Verlag, 1998.
9. J.-P. Hubaux, L. Buttyán, and S. Čapkun. The Quest for Security in Mobile Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking and Computing – MobiHOC 2001*, 2001.
10. HiperLAN2 Global Forum. *URL: http://www.hiperlan2.com*. Work in progress.
11. IEEE Computer Society LAN/MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - IEEE 802.11. 1997.
12. A. Jøsang. The right type of trust for distributed systems. *New Security Paradigms Workshop 1996*, ACM, 1996.
13. J. Jubin and J.D. Tornow. The DARPRA Packet Radio Network Protocol. *Proc. IEEE, vol. 75, no. 1*, 1987.
14. A. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.
15. Mobile Ad-hoc Networks (MANET).
URL: http://www.ietf.org/html.charters/manet-charter.html. Work in progress.
16. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks. *The 7th International Workshop on Security Protocols*, LNCS 1796, Springer-Verlag, 1999.
17. F. Stajano. The Resurrecting Duckling – what next? *The 8 th International Workshop on Security Protocols*, LNCS 2133, Springler-Verlag, 2000.
18. B. Warneke, M. Last, B. Leibowitz, and K.S.J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *Computer Magazine*, IEEE, Jan. 2001.
19. D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature 393*, 1998.
20. L. Zhou and Z.J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine, vol. 13, no. 6*, 1999.