IMPROVING SECURE DEVICE INSERTION IN HOME AD HOC NETWORKS

Olivier Heen, Jean-Pierre Andreaux, and Nicolas Prigent
Thomson R&D, Security Laboratory, Rennes, France
{Olivier.Heen; Jean-Pierre.Andreaux; Nicolas.Prigent}@thomson.net

Abstract Home ad hoc networks are sets of devices that self-configure and interact to offer enhanced services to their users. These networks are heterogeneous, dynamic and fully decentralized. Moreover, they generally lack a skilled administrator. These properties dramatically reduce the efficiency of classical security approaches: even defining the boundaries of such networks can be difficult. A way to solve this problem has been recently proposed, using the concept of secure long-term communities. This solution relies on one critical operation: the secure insertion of a device in the home ad hoc network. In this article, we propose two ways to improve this operation, using store-and-forward techniques. The first improvement deals with the ability to achieve insertion under loose connectivity circumstances. The other improvement deals with the ability for the user to use any trusted device in order to realize the insertion, especially in heterogeneous networks.

Keywords: Home Networks, Ad Hoc Networks, Security, Key-management.

INTRODUCTION

Communities are sets of principals linked by a trust relation. They can be encountered in many fields, from social interactions and commerce to corporate networks and home ad hoc networks.

Home ad hoc networks can be viewed as communities where principals are home devices (*figure 1*) and where trust relations are preconditions for device communication. A home ad hoc network may evolve along time: some devices could be inserted, while some others could be removed. Nevertheless, the home ad hoc network should still ensure consistency (a device is in the home ad hoc network or not), as well as correct trust management (a device can securely check if another one belongs to the home ad hoc network).

Y. Deswarte et al. (eds.), *Security and Protection in Information Processing Systems*

Due to their nature and environment, home ad hoc networks present additional constraints:

High heterogeneity
Devices have various computing powers, battery lifetimes, operating systems, communication protocols. Generally, not all the devices that constitute a home ad hoc network are able to communicate directly: see for instance the digital TV set and the modem in figure 1.

Erratic connectivity
A home ad hoc network may physically split and merge arbitrarily, according to the user's moves and devices availability. As showed in figure 2, the MP3 player, the digital assistant and the digital camera can for instance be temporarily disconnected from the rest of the network.

Poor administration
For wide public acceptance, home ad hoc networks should not be administration intensive for home users. Users may not have skills, motivation or time to perform advanced administrative tasks.

No central device
Users will acquire devices according to their needs. Therefore the existence of a specific device in the network cannot be assumed. In figure 1 for instance, no device is intended to interconnect all the other devices. Moreover, due to erratic connectivity, no device can be considered always available.

No central information
As some devices may be lost or stolen, trust management information should not rely on central information [9]. Consequently, solutions based on pre-shared secret data or centralized certification authority cannot be used.

One important point to ensure the security of a home network is first to define its boundary. For instance, technologies such as virtual private networks or firewalls suppose some notion of boundary or at least of an "inside" to be protected and an "outside" to be protected from. Consequently, before protecting the home ad hoc network, it is important to securely define which devices belong to it.

To achieve this, existing approaches such as [2] [5] [8] [13] [15] [16] rely on a secure insertion operation, generally with user participation. Other approaches, such as [17] do not require user participation, at the price of

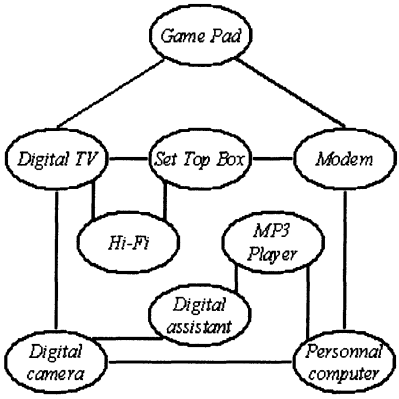

Figure 1. A home ad hoc network.

possible erroneous insertions. In this paper we propose generic improvements for robustness and handiness of the insertion operations with user participation.

In section 1, we indicate notations and recall from [13] a set of basic community operations, as well as their consequences on trust relations.

In section 2, we show how store-and-forward techniques can improve the robustness of the insertion operation.

In section 3, we show how store-and-forward techniques can improve the handiness of the insertion operation.

1. PRELIMINARIES

1.1 Notations

We mainly adopt notations from [13] and add explicit trust relations. Individual devices are denoted by letters. Trust relations are denoted by arrows: $x \rightarrow y$ means that device x trusts device y as being in its home ad hoc network. Mutual trust is denoted $x \leftrightarrow y$, meaning that $x \rightarrow y$ and $y \rightarrow x$ simultaneously hold.

Note that, in home ad hoc networks, trust relations between devices are transitive. Even if all the devices of the home ad hoc network do not necessarily belong to one and the same person, all the users of the same household share the same interest in interconnecting their devices securely. Consequently, they share the same policy about the devices that belong to the home ad hoc network, and all the devices that belong to it follow the same policy. In other words, when a device is inserted in a home ad hoc network, all the users agree on this insertion. Moreover, the devices that belong to the same home ad hoc network can reasonably

trust each others: when x trusts y as being in its home ad hoc network, all the devices of this home ad hoc network should also consider y as so. Transitive trust is denoted $x \rightarrow^* y$, which means that x trusts y as being in its home ad hoc network because a device z, that is trusted by x ($x \rightarrow z$), trusts or transitively trusts y ($z \rightarrow^* y$) as being in the home ad hoc network.

The set of devices that can bidirectionally communicate with x at a given time is denoted $\Phi(x)$.

The local community of x is denoted $\Lambda(x)$ and defined by $\Lambda(x) = \{y \mid x \rightarrow y\}$.

A community defined by its elements is denoted $\mathcal{C} = \{x_1 \dots x_n\}$ with $\forall x_i, x_j \in \mathcal{C}, x_i \rightarrow^* x_j$.

1.2 Basic operations

We now define the main operations on communities in home ad hoc networks. In this section, operations are defined supposing there is total connectivity between all the devices of the community. We show in section 2 how the insertion operation can be approximated under more realistic circumstances.

Initialization turns an isolated device into a community that contains only this device. Typically, initialization is done when acquiring a new device and turning it on for the first time.
$init : x \mapsto \{x\}$ with $\Lambda(x) = \{x\}$

Insertion adds a device to an existing community. For convenience, we suppose that the new device has been initialized just before insertion starts.
$insert : \{x_1 \dots x_n\}, \{y\} \mapsto \mathcal{C} = \{x_1 \dots x_n, y\}$ with $\Lambda(y) = \Lambda(x_i) = \mathcal{C}$

Merge is the extension of the insertion operation to communities.
$merge : \{x_1 \dots x_n\}, \{y_1 \dots y_m\} \mapsto \mathcal{C} = \{x_1 \dots x_n, y_1 \dots y_m\}$ with $\Lambda(x_i) = \Lambda(y_j) = \mathcal{C}$

Removal happens when a device has to leave the home ad hoc network, typically when it is sold or given. This device is considered available at the time of removal.
$remove : \{x_1 \dots x_n, y\}, y \mapsto \mathcal{C} = \{x_1 \dots x_n\}$ with $\Lambda(x_i) = \mathcal{C}$ and $\Lambda(y) = \{y\}$

Banishment is another form of removal, that occurs when a device that has to be removed from the community is not available, e.g. because it was lost or stolen.

$banish : \{x_1 \dots x_n, y\}, y \mapsto \mathcal{C} = \{x_1 \dots x_n\}$ with $\Lambda(x_i) = \mathcal{C}$ and $\forall i, x_i \not\rightarrow y$

2. ROBUST INSERTION

2.1 Realistic insertion conditions

In this section, we address the case of insertion under realistic circumstances, that is when perfect connectivity is not ensured. A home ad hoc network may physically split. In this case, it is made of physically separated partitions, and the devices of each partition cannot communicate with the devices of the others. Nevertheless, they still are in the same community. This happens for instance when the user temporarily leaves his or her home, taking some portable devices with him or her (*figure 2*). When she or he is back, all the devices are able to communicate altogether again.

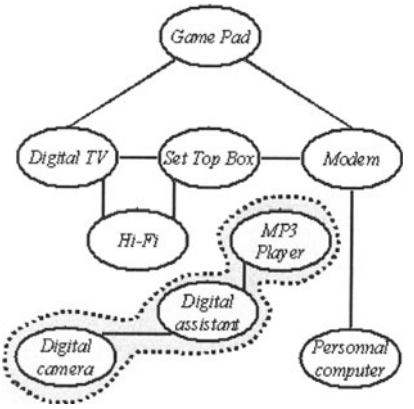

Figure 2. A partitioned home ad hoc network.

A partitioned community can be defined as: $\mathcal{C} = \{x_1 \dots x_n, y_1 \dots y_m\}$ with $\Phi(x_1) = \dots = \Phi(x_n)$ and $\Phi(x_i) \cap \Phi(y_j) = \emptyset$. The partition $\{x_1 \dots x_n\}$ is hereafter called the first partition. The rest of the community is hereafter called the other partitions.

When a user inserts a new device in the first partition of the community, the ideal insertion operation defined in section 1.2 is not possible. Only the devices in the first partition can receive the information that a new device is inserted. Devices from the other partitions are not reachable.

We propose to minimize this effect by using a secured and fully distributed store-and-forward mechanism of trust management information.

2.2 First stage: gaining trust

In our proposal, each device owns a public/private key pair. This key pair can for instance be set-up during the initialization operation. A device knows the other devices of its community through their public key, or a secure hash of their public key, as proposed in SUCV [10] and CAM [11]. We hereafter call the public key or the secure hash of a device its "provable identity".

We also choose that trust relation $\rightarrow$ is expressed by certificates. $x \rightarrow y$ stands for $Cert_x(y)$, meaning that the device x has issued a certificate for the device y's public key. Trust relation $\rightarrow^*$ is expressed by chains of certificates. $x \rightarrow^* y$ holds for the chain of certificates $1 \leq i \leq n, Cert_{x_i}(x_{i+1})$ with $x_1 = x$ and $x_n = y$.

Each device periodically broadcasts its provable identity to inform the other devices of its physical availability.

We now indicate the algorithm for inserting y in the community of x.

1 x detects the incoming device y by receiving its provable identity.

2 x notifies the user, displays the provable identity of y using a secure and user friendly representation such as a random art visualization [12], and waits for user confirmation.

3 The user checks that the provable identity displayed on x's screen is really y's provable identity (for instance by asking y to display a random art visualization of its own provable identity). If so, the user confirms to x that y belongs to x's community. x then generates a certificate $x \rightarrow y$, and stores it.

4 x sends the certificate $x \rightarrow y$ to y.

5 y stores the certificate $x \rightarrow y$ if and only if it already knows $y \rightarrow x$.

6 y answers by a certificate $y \rightarrow x$ if and only if it already knows $y \rightarrow x$ or $y \rightarrow^* x$.

7 If x received the answer, x stores the certificate $y \rightarrow x$.

8 If x knows the certificate $y \rightarrow x$, then x notifies the user that the insertion of y is complete.

Here is the explanation that this algorithm correctly achieves insertion, provided that the user confirms to each device that the provable identity it displays matches the one to be inserted.

During the insertion, both devices apply the same algorithm. Without loss of generality, we suppose that the user first logs on device x. He or she sees the notification that a new device y is waiting for insertion. The user checks the random art representation of y's provable identity. Consequently, no other public key than y's can be inserted. The user then confirms. At this time, y has no information about x. In particular, y does not know that $y \rightarrow x$, and will not reply to any message from x. Then, at the end of step 4, x only learned $x \rightarrow y$, and y learned nothing.

Now, the user logs on y to complete the insertion. y executes the same algorithm, with x knowing that $x \rightarrow y$. Thus, x answers to y message on step 6, providing y more information on trust relation. At the end of step 8, x learned $y \rightarrow x$ (x already knew $x \rightarrow y$) and y learned $y \rightarrow x$ and $x \rightarrow y$. In other words, both x and y learned that $x \leftrightarrow y$. □

At this stage, robustness comes from the fact that a device never assumes that sent messages are received. Instead, the user validates insertion by giving confirmation to the first device and then to the second device.

2.3 Second stage: spreading trust

In order to reduce user's involvement and to spread trust, the devices of the same community exchange information about the other devices that belong to it.

On a regular basis, a device x sends to each device y for which it has the certificate $x \rightarrow y$ and the other one $y \rightarrow x$ all the certificates it has about the devices z_i that x trusts as being in the community (i.e. the certificates $x \rightarrow z_i$). It also sends the certificates or the chains of certificates it has that prove that these z_i trust x (i.e. $z_i \rightarrow x$ or $z_i \rightarrow^* x$), as explained in the following algorithm:

1 x detects y, for which it has a mutual trust relation with, i.e. for which it has both certificates $x \rightarrow y$ and $y \rightarrow x$.

2 x sends to y all the certificates it has about the devices z_i that it trusts as being in the community, i.e. the certificates $x \rightarrow z_i$, as well as the certificates or the chains of certificates it has that prove that these z_i trust x, i.e. $z_i \rightarrow x$ or $z_i \rightarrow^* x$.

3 If y knows $y \rightarrow x$ and $x \rightarrow y$, for each z_i that y does not know as $y \rightarrow z_i$, y generates the certificate $y \rightarrow z_i$ and stores it. It also appends the certificate $x \rightarrow y$ to the certificate $z_i \rightarrow x$ or the chain

of certificates $z_i \rightarrow^* x$. y stores this chain of certificates $z_i \rightarrow^* y$. y now knows that z_i belongs to the community, and can prove it the next time y and z_i meet, as explained later.

4 If y knows $y \rightarrow x$ and $x \rightarrow y$, for each z_i that y does know as $y \rightarrow z_i$, it checks the length of the certificate chain obtained by appending the certificates $z_i \rightarrow x$ or the chains of certificates $z_i \rightarrow^* x$ to the certificate $x \rightarrow y$. If it is shorter than the chain of certificates it already has for z_i, it replaces it by the new one.

At the end of the algorithm, y knows all the devices z_i that x trusts as being in the community. Moreover, y has all the chains of certificates to prove z_i that they belong to the same community.

When y detects an incoming device z_i that it trusts, i.e. for which is has a certificate $y \rightarrow z_i$, and for which it does not have the certificate $z_i \rightarrow y$ but a chain of certificates $z_i \rightarrow^* y$, y sends to z_i the chain of certificates $z_i \rightarrow^* y$ to prove to z_i that they belong to the same community, as explained in the following algorithm:

1 y detects an incoming device z_i, y knows that $y \rightarrow z_i$, and y has a chain of certificate $z_i \rightarrow^* y$.

2 y sends to z_i the chain of certificate $z_i \rightarrow^* y$.

3 If the received information $z_i \rightarrow^* y$ is correct, i.e. if the chain of certificates is valid, z_i computes the certificate $z_i \rightarrow y$, stores it, and sends it to y.

At the end of the algorithm, both y and z_i know they are in the same community, and each of them knows that the other one knows it.

2.4 Related Work

Some proposals have already been made for secure group management in ad hoc networks. Zhou and Haas [18] have proposed to use threshold cryptography to certify public keys in ad hoc networks. Some special devices, known as servers, have shares of a private key corresponding to a public key known by all the devices belonging to the group. A device is inserted when a minimum number of servers generates a partial signature of its public key. This proposal requires a certain number of servers to be present to insert a new device, and consequently does not perfectly fit with the requirements we stated for home ad hoc networks.

Stajano [14] proposes to manage groups of devices by uploading policy rules during the imprinting phase of the Resurrecting Duckling [15] about

the devices that should be trusted. However, the knowledge of the trusted devices is obtained only at the time of imprinting, and nothing is proposed to update it. Some other proposals [6] [1] use and extend [15] to manage groups, but none of them proposes updates of the groups that fit with the requirements we stated for home ad hoc networks.

Other works have also proposed to use web of trust in order to set up trust relations between principals in an ad hoc network. Because there is no always-available-PKI in ad hoc networks, Capkun, Buttyan and Hubaux [3] [7] [4] propose that each device stores a subset of the available certificate. Two devices that want to securely communicate share the certificates they know and try to find a chain of certificates that could certify each other's public key. These proposals differ from ours because the goals are different. While Capkun, Buttyan and Hubaux intend to create pairwise secure relations between devices, we intend to securely manage groups. Consequently, the way each device manages its certificates is different.

3. HANDY INSERTION

Aside from robustness, another essential criterion in home ad hoc networks is handiness. If insertion is not handy enough, users will eventually try to circumvent boring security mechanisms related to it. In section 3.1, we recall how the user is involved in most existing insertion methods. In section 3.2, we modify the proposed mechanism to make the insertion less demanding to the user. In section 3.3, we discuss some consequences and extensions to the *merge* operation.

3.1 State-of-the-art

Different proposals [2] [5] [8] [13] [16] have been made to securely insert a device in a group of devices. Because they come along with their own notion of community and their own definition of "secure insertion", all these proposals are hardly comparable. Moreover, not all these proposals fit home ad hoc networks, first because devices are not always available in home ad hoc networks, and also because devices cannot always communicate by pair, mostly due to the fact that the devices in home ad hoc networks may use heterogeneous communication protocols and media. Nevertheless, some common properties can be drawn out:

User participation
: Generally, both the insert*ed* device and the insert*ing* device require user actions such as entering a PIN code or a password, pressing a button, generating a key and inserting it in the right devices, etc.

Auto-detection

To reduce user participation, incoming devices announce themselves to, or are detected by, at least one device in the community.

Need for secure channel

Most insertion methods require transmission of secret information, either between devices or between their users, e.g. whispering a password.

It also appears that, whenever auto-detection is used, the user is not given the possibility to choose the inserting device.

Instead, the detector may become the inserting device (*figure 3*). This case could happen for instance with the Resurrecting Duckling [15] and with 802.11 [8] in ad hoc mode.

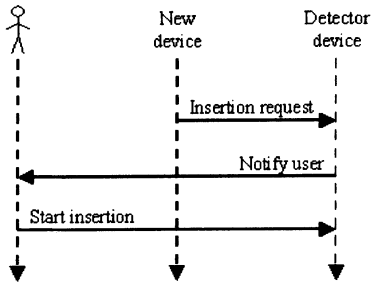

Figure 3. The detecting device immediately becomes the inserting device.

Another option is that the detector forwards information to a predetermined controller that becomes the inserting device (*figure 4*). This case could happen for instance with Zigbee [5] [9], UPnP [16] and 802.11 [8] in infrastructure mode when the access point is used as a controller. Controller based solutions generally correspond to centralized solutions, that do not meet home ad hoc networks requirements.

Auto-detection should not imply forced choice of the inserting device. First of all, this complicates the user's understanding of the process when he or she is surprised by the automatic choice. Moreover, this forbids insertion when the user does not have enough rights to log on the detector or controller device, even though he or she possesses enough rights to log on other devices of the community that could have endorsed the role of inserting device.

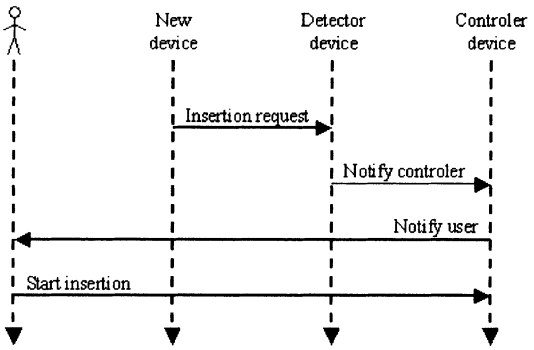

Figure 4. The detecting device forwards to a controller, that becomes the inserting device.

3.2 Free choice of the inserting device

We now modify the beginning of the insertion method described in section 2 so that the user freely chooses the inserting device. In order to do so, we propose to use a store-and-forward technique: devices of the community store insertion requests instead of immediately serving them. The user freely chooses a device in the community, and asks to start insertion. This device asks all the reachable devices of its community for their stored pending requests, and then endorses the next steps of the insertion process, on behalf of the formerly detector devices.

More precisely, if the user chooses x, then x sends a message to all the devices in $\Lambda(x)$ that are reachable, including itself. Each of these devices that has pending requests sends them back to s. If at least one pending request is collected, the chosen device can execute its usual insertion protocol, playing the role of the inserting device. When modifying the beginning of the insertion process, one should care about some technical details:

- Once a detector has sent its pending requests, it should forget them immediately in order to avoid retransmission of formerly honored requests.

- When the chosen device endorses the role of inserting device, there is no need to notify the user since he or she is already logged on, as done in step 2 of the algorithm in section 2.2.

The beginning of the algorithm explained in section 2.2 has to be modified as follows:

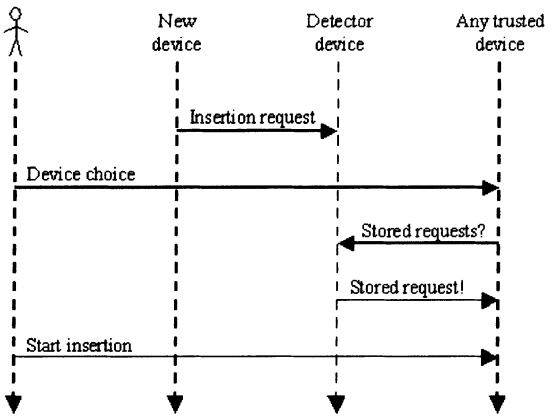

Figure 5. A new method for starting device insertion.

1 The user chooses a device x of the community, logs in, and asks for extended insertion.

2 x securely sends to all the reachable devices $\{y_0 \ldots y_n\}$ of its community a request for pending request.

3 Each device y_i broadcasts the provable identity of x and listens for provable identities they do not know as being in the community.

4 Each device y_i sends to x the list of provable identities it has stored.

5 The algorithm described in section 2.2 continues normally at step 2. The only difference is that x sends the messages to the device that has returned to it the provable identity to be inserted. This device forwards the messages between x and the device to be inserted.

It is important that each device y_i broadcasts the provable identity of x. Indeed, if it does not, the device to be inserted in the community can not receive the provable identity of x, and consequently can not insert x in its own community.

3.3 Consequences

This proposal can be applied to many insertion method, since only the beginning of the process needs to be modified.

The user finds more comfort in choosing his or her favorite device for performing insertion. Moreover, insertion will benefit from rich user

interactions such as checking the random art visualization [12] of a provable identity instead of its hexadecimal string, key generation could benefit from mouse moves, etc. If at least one device in the community can perform such functions and make its life easier, the user will intuitively choose it.

An interesting point is that this extension of the insertion of a device in a community can also be used for the merge of two communities. Consequently, none of the detecting devices need to be used, (*figure 6*) and the user can freely choose the inserting device in both the communities to be merged. Technical details of this extension are not discussed in this paper.

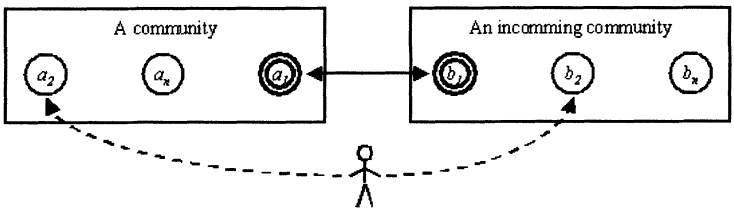

Figure 6. Unconstrained device choice when merging communities.

CONCLUSION

Home networks are a very demanding kind of communities. Most difficulties come from the mix of security needs and poor administration. In this paper, we've proposed ways to improve robustness and handiness of one critical home ad hoc network operation: the secure insertion of a device. We believe that using these proposals in home devices will allow better security acceptance from the user. Further work in the same direction should now consider extensions such as: allowing insertion of guest devices within the community or allowing temporary merging of communities.

Acknowledgments

The authors would like to thank Refik Molva and the anonymous reviewers for their insightful remarks and comments. They also would like to thank Alain Durand and Jean-Louis Diascorn for their valuable remarks and support.

References

[1] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, Feb. 2002.

[2] Bluetooth Inc. Bluetooth core specification 1.1, 2001.

[3] S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1), January-March 2003.

[4] S. Capkun, J. P. Hubaux, and L. Buttyan. Mobility helps security in ad hoc networks. In *Proceedings of the Fourth International Symposium on Mobile Ad Hoc Networking and Computing*, 2003.

[5] E. Callaway et al. Home networking with IEEE 802.15.4 : a developing standard for low rate WPAN. *IEEE Communications Magazine*, pages 70–76, Aug. 2002.

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

[7] J. P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, October 2001.

[8] IEEE Standard Department. IEEE 802.11 Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY), 1999.

[9] T. S. Messerges, J. Cukier, T. A. Kevenaar, L. Puhl, R. Struik, and E. Callaway. A security design for a general purpose, self-organizing, multihop ad hoc wireless network. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Oct. 2003.

[10] C. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) identifiers and addresses. In *NDSS'02*, Feb. 2002.

[11] G. O'Shea and M. Roe. Child-proof authentication for MIPv6 (CAM). *ACM SIGCOMM Computer Communication Review*, 31(2):4–8, 2001.

[12] A. Perrig and D. Song. Hash visualization: a new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, pages 131–138, 1999.

[13] N. Prigent, J.-P. Andreaux, C. Bidan, and O. Heen. Secure long term communities in ad hoc networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Oct. 2003.

[14] F. Stajano. The Resurrecting Duckling – what next? *Lecture Notes in Computer Science*, 2133:204–211, 2001.

[15] F. Stajano and R. Anderson. The Resurrecting Duckling: Security issues for ad-hoc wireless networks. In *7th International Workshop on Security Protocols*, pages 172–194, 1999.

[16] The UPnP Initiative. The UPnP Forum, *www.upnp.org*, 2004.

[17] V. Cahill et al. Using trust for secure collaboration in uncertain environments. *Pervasive Computing Mobile And Ubiquitous Computing*, 2(3), July-September 2003.

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