IMPROVING SECURE DEVICE INSERTION IN HOME AD-HOC NETWORKS

Olivier Heen, Jean-Pierre Andreaux, Nicolas Prigent
Thomson R&D, Security Laboratory, Rennes, France

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

Keywords: Network Security, Key-management.

Introduction

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

Communities may evolve with time: new devices may demand trust, others can get distrusted... while still offering consistency: a device is in the community or not, as well as correct trust management: a device can check trust with other devices.

Home ad-hoc networks can be viewed as communities where principals are home devices (*figure 1*) and where trust relations serve as preconditions for device communication.

Due to their nature and environment home ad-hoc networks are subject to additional constraints:

High heterogeneity
: Devices have various computing power, autonomy, 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 arbitrary split and merge, according to user moves and devices availability. Nevertheless, trust relations shall be enforced as best as possible. In figure 1 the MP3 player, digital assistant and digital camera can disconnect from the rest of the network.

Poor administration
: For wide public acceptance, home ad-hoc networks shall not be too demanding to home users. Users may not have skills, motivation or time left 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 designed to attach all other devices.

No central information
: As some devices may be lost or stolen, trust management information should not rely on centralized information. This discards solutions based on pre-shared secret data or centralized certification authority.

One important question when considering communities in home environment is:

Can trust relations always be set-up, enforced and managed while respecting all the identified constraints?

This question, also referred to as "the home ad-hoc network boundary problem" is the mandatory prerequisite before even thinking to securize home ad-hoc networks (technologies such as virtual private networks, firewalls or intrusion detection systems suppose some notion of boundary or at least an "inside" to be protected and an "outside" to be protected from).

Figure 1. A home ad-hoc network.

Existing approaches such as [1, 3, 5, 2, 7, 6] heavily rely on a secure insertion operation, generally with user participation (a recent approach from [4] does not require user participation, at the price of possible false insertion). 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 [1]) a set of basic community operations that do preserve trust relations.

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

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

1. Preliminaries

1.1 Notations

We mainly adopt notations from [1] 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. Mutual trust is denoted $x \leftrightarrow y$, meaning that $x \rightarrow y$ and $y \rightarrow x$ simultaneously hold. Note that trust relations are transitive.

The set of devices that can bi-directionally communicate with device x is denoted $\Phi(x)$.

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

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

1.2 Basic operations

We now give a definition of the main operations over communities. At first, operations are defined under ideal conditions, when there is perfect connectivity between all devices in the community. We show in section 2 how the ideal insertion operation can be approximated under more realistic circumstances.

Initialization turns an isolated device into a community that contains only this device. Typically, this is done when acquiring a new device.
$init : x \mapsto \{x\}$ with $\Lambda(x) = \{x\}$

Insertion adds one device to an existing community. For convenience, we suppose that the new device is 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 $\Lambda(x_i) = \Lambda(y_j) = \mathcal{C}$

Remove is the converse of insertion. Typically, this is done when re-selling or giving devices. The concerned device must be 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 $\Lambda(y) = \emptyset$

Banish is another form of removal. Typically, this is done when a device was lost or stolen and thus unavailable for remove operation.
$banish : \{x_1 \dots x_n, y\}, y \mapsto \mathcal{C} = \{x_1 \dots x_n\}$ with $\Lambda(x_i) = \mathcal{C}$ and $\forall i, x_i \not\rightarrow y$

2. Robust insertion

2.1 Realistic insertion conditions

We now address the case of insertion under realistic circumstances, that is when perfect connectivity is not ensured. In home ad-hoc networks, this currently happens when the user leaves its home, taking some portable devices with him/her (*figure 2*).

In the general case, 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 part $\{x_1 \dots x_n\}$ is called the first connected part. The rest of the community is called the other connected parts.

Figure 2. A partitioned home ad-hoc network.

What should happen when a user inserts a new device z into the first connected part of the partitioned community?

Even before describing the method, one thing is clear: the ideal insertion operation as defined in 1.2 is not possible in this case. Only devices from the first connected part can receive the information that a new device was inserted. Devices from the other connected parts are not reachable.

We propose to minimize this effect by the use of secured and fully distributed store-and-forward of trust management information.

2.2 First stage: gaining trust

We first indicate the general algorithm for inserting device y within the community of device x.

1 x detects incoming device y.

2 x notifies users and waits for confirmation.

3 x learns $x \rightarrow y$.

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

5 y learns $x \rightarrow y$ if and only if it already knows $y \rightarrow x$.

6 x asks to y if it knows $y \rightarrow x$.

7 y answers $y \rightarrow x$ if and only if it already knows $y \rightarrow x$.

8 If answer is received, x learns $y \rightarrow x$.

9 If x knows $y \rightarrow x$ then x notifies user that insertion of y is finished.

Here is the proof that this algorithm correctly achieves insertion, provided that users confirms on each device.

During the insertion, both devices apply the same algorithm. Without loss of generality, we suppose that the user first logs on device x. S/he will see the notification that a new device y is waiting for insertion, and then give confirmation. At this time, y has absolutely no information concerning x. In particular, y doesn't know $y \rightarrow x$, and wont reply to any message from x. Then, at the end of the 9 steps, x only learned $x \rightarrow y$, and y learned nothing.

Now, user logs on y to complete the insertion. y executes the same algorithm, with x knowing something about y, namely: $x \rightarrow y$. Thus, x will answer y messages, providing y more information on trust relation. At the end of the 9 steps, 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 the information $x \leftrightarrow y$. □

At this stage, robustness mostly comes from the fact that device never assume that sent messages are received. Instead, this is the user who validates information exchanges while giving confirmation on first device and then on second device.

2.3 Second stage: spreading trust

Once a device of the community trusts a new device, it should spread this information to all other devices of the community.

We indicate the behavior of a device x in presence of a trusted device y:

1 x detects incoming device y, and x knows $x \rightarrow y$ (or *a fortiori* $x \leftrightarrow y$).

2 x sends to y every information of the form $z \rightarrow^* x$ in its possession.

3 upon reception, y uses information $z \rightarrow^* x$ and $x \rightarrow y$ to compute $z \rightarrow^* y$.

Two special cases may arise during step 3.

3.a y already knows information of the form $z \rightarrow^* y$, in this case, y may forget the longest chain.

3.b if $z = y$, that is if y received information $y \rightarrow^* x$, then y computes $y \rightarrow x$ and sends this information back to x.

Chains of information of the form $z \rightarrow^* x$ allows storage and sending of partially build trust relations. Moreover, using their local knowledge,

devices may sometime reduce chain length (case 3.a) or even finalize insertions with no need of user confirmation (case 3.b).

2.4 Implementation notes

A straightforward implementation of the algorithms described in 2.2 and 2.3 is obtained doing the following:

- Each device owns a public / private key pair. This can be set-up during the initialization operation.
- Trust relation $\rightarrow$ is enforced by certificate relation. $x \rightarrow y$ stands for $Cert_x(y)$, meaning that device x certified decice y public key.
- Trust relation $\rightarrow^*$ is enforced by certificate chain relations. $x \rightarrow^* y$ stands for the chain $1 \leq i \leq n, Cert_{x_i}(x_{i+1})$ with $x_1 = x$ and $x_n = y$.
- In order to allow detection, devices can simply broadcast their public key, or even a secure hash of their public key.

3. Handy insertion

Aside from robustness, another essential criterion in home ad-hoc networks is handiness.

In 3.1 we recall how user is involved in most existing insertion methods. In 3.2 we propose a slight modification that renders insertion less demanding to the user. In 3.3 we discuss some consequences and extension to the *merge* operation.

3.1 State-of-the-art

There exist many different solutions to securely insert devices in communities. Some are at research stage such as the resurrecting duckling [3] from F. Stajano and R. Anderson, or a recent proposal [1] from the authors with N. Prigent and C. Bidan. Other solutions already exists within standards such as UPnP [6], Bluetooth [2], 802.11 with activated security features [5] or Zigbee [7].

Because they come along with their own notion of community and their own notion of what "secure insertion" means, all these solutions are hardly comparable. Moreover, not all solutions will fit in environment such as home ad-hoc networks, mostly because of heterogeneity.

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, pressing a button, generating a key...

Auto-detection
To minimize 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 device 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 may happen with resurrecting duckling and the method from [1], with 802.11 in ad-hoc mode and with Bluetooth, depending of user configuration choices (Bluetooth offering many possibilities).

Figure 3. Detecting device immediately becomes inserting device.

We believe that auto-detection should not imply forced choice of the inserting Another option is that the detector forwards information to a predetermined controller that becomes the inserting device (*figure 4*). This case may happen with Zigbee, UPnP and 802.11 network in infrastructure mode when the access point is used as a controller. Note that controller based solutions correspond to centralized solutions that generally do not meet home ad-hoc networks requirements: controller availability can not be assumed.

Auto-detection should not imply forced choice of the inserting device. At first, this can complicate user understanding of the whole process

Figure 4. Detecting device forwards to a controller, that becomes inserting device.

when s/he is surprised by the automatic choice. Moreover this can forbid insertion when the user does not have enough rights to log on the inserting device, even thought s/he possess enough rights to log on other devices of the community (that could have endorsed the role of inserting device).

3.2 Free choice of the inserting device

We now modify the beginning of the insertion method of 2 so that user may always choose the inserting device. In order to do so, we use a store-and-forward technique: devices of the community will store insertion requests instead of immediately serve them. They will do so until the user freely chooses one device in the community. This device will inquire for stored pending requests and then endorse the next steps of the insertion process, on behalf of the receiving devices.

More precisely, if the user chooses device a, then a sends a message to all devices in $\Delta(a)$, including itself. If at least one pending request is collected, the chosen device can follow 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:

- If many pending requests are collected, they should be presented with no ambiguity, for instance sequentially, to the user so that s/he does not confound requests from several new devices.

- 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 s/he is already logged on.

Figure 5. A new method for starting device insertion.

3.3 Consequences

This proposal should apply to any existing insertion method, since only the beginning of the process needs to be modified.

User will find more comfort to choose its favorite interface for performing insertion. Moreover, some insertion methods will benefit from rich user interactions such as checking of a random art [8], using mouse moves for key generation... If at least one device in the community can perform such functions, user will intuitively choose it.

The extension of the method to the merge of whole communities led us to a surprising result: whenever devices of two communities discover each other, the modified insertion process may start, letting the user free to chose any inserting device in each community. Then, none of the detecting device need to be used (*figure 6*). This seems to allow insertion in complex situations, when multiple users try to merge communities (authors are currently investigating this way).

Conclusion

Home networks are very demanding kinds of communities. Most difficulties come from the mix of security needs and poor administration.

Figure 6. Unconstrained device choice when merging communities.

By improving robustness and handiness of secure insertion algorithms we seek better public acceptance of home ad-hoc networks solutions, and better overall security. Pursuing this way, we should improve other operations, such as removal or banishement. Note that, in some sense, these operations are simpler since they only involve communication between *already* trusted devices.

Acknowledgments

The authors would like to thank Nicolas Prigent and Alain Durand for valuable remarks and support.

References

[1] N. Prigent, J.P. Andreaux, C. Bidan, O. Heen, *Secure Long Term Communities In Ad Hoc Networks*, 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), Oct 2003.

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

[3] 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.

[4] V. Cahill, et al., *Using Trust for Secure Collaboration in Uncertain Environments*, in Pervasive Computing Mobile And Ubiquitous Computing, vol. 2(3), July-September, IEEE, 2003.

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

[6] The UPnP Initiative, *The UPnP Forum*, `www.upnp.org/`.

[7] Ed Callaway et al. *Home networking with IEEE 802.15.4 : a developing standard for low rate WPAN*, IEEE Com. Mag., pp. 70–76, Aug 2002.

[8] A. Perrig and D. Song, *Hash Visualization: a New Technique to improve Real-World Security*, International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99), pp. 131–138, 1999,