A Key-Distribution Scheme for Wireless Home Automation Networks

Arno Wacker, Timo Heiber, and Holger Cermann
Institute for Parallel and Distributed Systems (IPVS)
Universität Stuttgart, Stuttgart, Germany
{wackerao | toheiber | cermanhr}@hermes.informatik.uni-stuttgart.de

***Abstract*— Wireless home automation networks are one example of how wireless technologies may soon become part of our daily life, yet security in existing products is woefully inadequate. An important problem in this area is the question of secure key distribution. In this paper we present a key-distribution scheme geared towards home automation networks, but applicable to other networks with related properties as well. Our approach uses a decentralized scheme that is designed to work on resource-poor devices, allows easy addition and removal of devices and limits the workload on the end user while guaranteeing the secrecy of the exchanged keys even in the presence of subverted nodes.**

I. Introduction

In the near future, many common devices at home will have computational power and wireless communication capabilities. One of the possible applications are wireless networks for home automation. Imagine a private home equipped with motion-, light-, temperature- and other sensors and actuators for opening the door, dimming the light, controlling the heating and so on. These sensors/actuators can be used in order to enhance the individual's lifestyle, e.g. the heating is turned on automatically when the owner of the house comes home and the light is switched on in rooms where motion is detected. Although there are existing solutions for home automation (i.e. [1], [2], [3]), most of them do not address security aspects in an appropriate way (if at all). At the same time, security in those systems is a crucial factor since having these new technologies at home provides many new ways for adversaries to invade an individual's personal life. An example of an attacker that would be motivated to do so might be a thief that wishes to gather information about when or if there is somebody at home.

In order to counteract this threat, we need to secure the communication with mechanisms that provide secrecy, authenticity, integrity and freshness of messages. Encrypting the communication will provide secrecy while integrity can be provided by the use of secure hash functions and freshness by the use of counters. For authenticity, either digital signatures or pairwise unique shared keys will be needed. It follows that the main need is to place unique encryption keys on the devices and that these keys must be distributed over a secure channel.

In order to do so, we emphasize a decentralized solution, i.e. the devices can establish keys without referring to a central authority. This approach avoids a single point of trust, thus even when a device is subverted by an attacker, the key exchange for the remainder of the network will remain secure. In this work, we present a parametrized solution which guarantees the secrecy of a key exchange as long as there are less than s subverted devices, where s can be chosen according to the actual security requirements.

The remainder of this paper is organized as follows: Section II introduces the properties of home automation networks. With theses properties in mind we discuss in section III existing approaches for key exchange in wireless and ad hoc networks. Section IV will then describe our system and adversary model while our approach will be presented in section V. The correctness of our approach will be proven in section VI, and section VII discusses the properties of our approach and compares them to the requirements. We present an extension to our scheme in section VIII and conclude in section IX with some thoughts about future work.

II. Properties of Home Automation Networks

Typical home automation networks consist of many small devices (sensors and actuators) which, in order to perform their automation tasks securely, need to communicate in a secure way.

The devices of such an network must be inexpensive. This gives rise to two problems: There is no way to make the devices absolutely tamper resistant [4] and the devices will only have limited resources. An example controller for such a device might be the Atmel AT90S2313, whose memory is limited to 512 byte RAM and 2 kbyte flash memory, only part of which will be available for security purposes.

With the above properties of home automation networks in mind, the following requirements can be stated:

- The key distribution scheme must be decentralized – it should not rely on the tamper resistance or theft protection of any single device.
- Since we are dealing with mostly resource-poor devices, asymmetric cryptography is problematic [5]. Hence, we propose to use symmetric cryptography (see the next section for alternative approaches).

Another aspect of home automation networks is that these networks usually grow over time when the user buys additional devices. Also, some older device might get removed or replaced. Therefore, any security solution for such a network must allow to add or remove devices and the corresponding keys easily.

Lastly, we also emphasize the need for a solution that is easy to install and use for the end user.

III. Related Work

Existing approaches for home automation can be divided in two categories according to the communication media used. The first one is based on wired networks ([1], [3]) and the second on wireless networks ([2], [3]). None of the systems we analyzed provide security in any meaningful way, i.e. these systems are completely open to an attacker who has access to the communication media. Therefore, especially the wireless technologies are extremely vulnerable.

Some solutions for this problem exist for wireless ad hoc networks or sensor networks. Such networks usually can be pre-configured and it is a priori known how big the network is, or how big it might get. These approaches usually do not address the issue of easy addition or removal of devices.

The existing solutions for wireless ad hoc networks can be grouped in four different categories, based on whether they employ symmetric cryptography or asymmetric cryptography, and whether they follow a centralized or a decentralized approach.

a) Centralized asymmetric approaches: A straightforward solution is to use certificates issued by a central certification authority. A central server thereby stores and signs the public keys of the individual devices. The major problems with this approach are the limited resources of the devices and the need for a central (potentially vulnerable) authority itself.

b) Decentralized asymmetric approaches: Since a centralized authoritative device is a single point of failure, asymmetric decentralized approaches have been proposed. Such designs can be achieved by distributing the certification authority over the participating devices [6], [7].

Both asymmetric approaches share a common problem: If for instance small sensor devices are included – even with the use of supporting high performance nodes [8] – the use of asymmetric cryptography is often not possible due to delay and energy constraints. On a Palm Pilot, for example, the generation of a signature with a 1024 bit RSA key requires approximately 36 seconds [5].

c) Centralized symmetric approaches: Due to the difficulties of the asymmetric approaches, the use of symmetric cryptography is proposed in many recent publications. Similar to the asymmetric approaches, one possible solution is the use of a centralized server, acting as an intermediary for pairwise secure channel establishments [9]. The common centralized symmetric approach for fixed networks would be Kerberos [10].

d) Decentralized symmetric approaches: The final approach, and the one that most appropriately addresses our requirements, is the use of symmetric cryptography in a decentralized fashion. To the best of our knowledge, the following approaches have been published:

A simple approach was shown by Basagni et al. in [11]. However, their approach assumes tamper-resistant devices, a notion that we consider problematic in the present context.

A different approach was examined by Chan et al. [12]. This approach uses a randomly predistributed set of keys. The authors present several schemes that allow to establish shared keys after deployment of the devices. The main drawback of their design is that due to the random predistribution of keys, two arbitrary nodes might not be able to establish a shared key at all, i.e. the design cannot guarantee the key establishment functionality.

The third decentralized symmetric approach by Zhu et al. [13] – published concurrently to [12] – proposes a similar scheme. It also uses a predistributed set of random keys. This way, devices cannot be integrated in a network without preparing it using a common programming device. Other than [12], Zhu et al. are proposing a pairwise key establishment protocol using multiple logical paths, as proposed in [14]. This way, a key can be split over multiple untrusted paths and resistance against subverted nodes is improved. Due to the random predistribution, the actual existence of different paths in the network is not assured in any way.

IV. System and Adversary Models

Our network consists of independent devices, communicating over a wireless channel. The channel itself is insecure, i.e. anyone can listen and send to the channel. The number of nodes is not predetermined or constrained in any way, as it may change due to the introduction of new devices to the network or node failures. We assume a non-partitioned network, therefore communication between two arbitrary nodes is always possible.

Since we consider the use in a home environment, we make the assumption that an attacker might physically manipulate a new or already integrated device. Since tamper resistance is in principle hard to achieve [4], and due to the reasons discussed in section II, we need to take the possibility into account that a small number of devices gets subverted by an attacker.

We categorize attackers (or subverted devices) in two classes:

1) The *eavesdropping attacker*. This attacker is only interested in learning about secrets on other devices, e.g. newly established keys. The objective of the attacker is to eavesdrop on communications between other devices.
2) The *denial-of-service attacker*. This attacker acts in order to cease the functionality of the network. Note that we do not consider denial of service on the physical layer (e.g. jamming the frequency).

When a device gets subverted we assume for the purpose of this work that the attacker is primarily interested in eavesdropping.

V. Our Approach

Overall, our objective is to establish a shared key between any pair of devices in the network, without giving an eavesdropping attacker the possibility to learn the newly established key. Additional objectives are ease of setup from a user's

standpoint and the ability to cope with the limited amount of key storage space on the nodes.

In order to set up the home automation network (and introduce new devices to it), we follow the principle of physical contact [15]. Such physical contact establishes a new shared key between two devices. Obviously it is impractical to establish physical contact between each pair of devices in the network, both in practical terms and in terms of the size of the necessary key storage. Hence, it will be necessary to limit the number of physically exchanged keys and establish additional shared keys between devices as necessary.

A. Network Model

Before we describe our approach in detail, we define a formal representation of our network, as follows:

A network is represented as an undirected graph $G = (V, E)$, where V is the set of devices in the network, and E represents the set of shared keys between devices where $\{v_1, v_2\} \in E$ iff the nodes v_1 and v_2 share a symmetric key. We will use the term *device* to indicate the physical device and the term *node* to indicate the representation of that device in the graph.

B. Setting up the Network

In order to set up the network and whenever a new device is added to the network, a certain number keys has to be exchanged through physical contact. We require that each device that is added to the network shares a physically exchanged key with at least s devices that are already part of the network and refer to s as the *security level* of the network. The actual procedure is given in algorithm 1, using the formal representation of the network.

Per algorithm 1, for the first $s+1$ nodes, the network will be represented by a fully connected graph and for each additional node introduced, a key needs to be physically exchanged with s other devices in the network, i.e. s new edges from the new node to previously existing nodes will be added.

Algorithm 1 Introducing a new node to the graph

1: Given a graph $G = (V, E)$ with $n = |V|$ with nodes $v_i \in V$ and $e_i \in E$ and a new node v_{n+1}
2: $V := V \cup \{v_{n+1}\}$;
3: **if** $s \geq n$ **then**
4: {device corresponding to v_{n+1} creates n new keys}
5: **for** $i = 1$ to n **do**
6: $E = E \cup \{v_{n+1}, v_i\}$;
7: {establish a new key through physical contact}
8: **end for**
9: **else**
10: $V' :=$ random subset of $V - \{v_{n+1}\}$ with $|V'| = s$;
11: {device corresponding to v_{n+1} creates s new keys}
12: **for** $i = 1$ to s **do**
13: $E = E \cup \{v_{n+1}, v_i\}$ with $v_i \in V'$;
14: {establish a new key through physical contact}
15: **end for**
16: **end if**

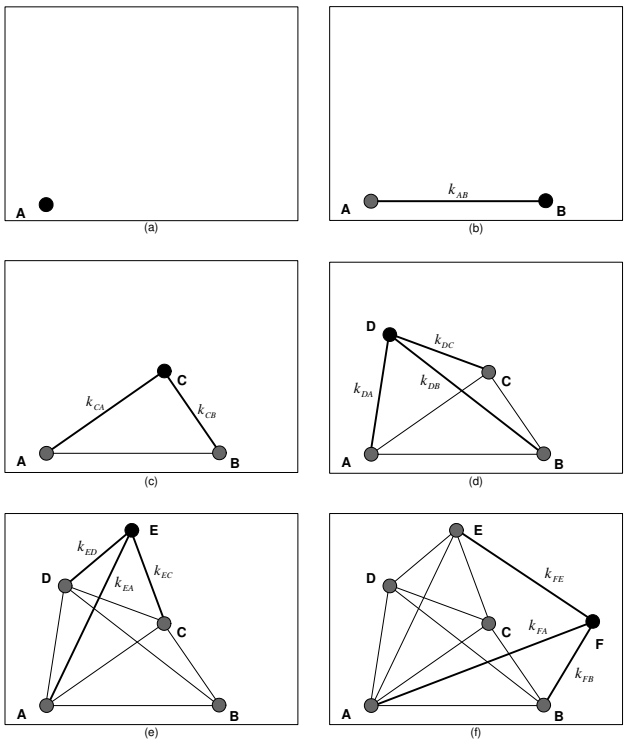

Fig. 1
INTRODUCING NEW NODES TO THE NETWORK GRAPH ($s = 3$)

For example, consider fig. 1: For a desired security level of $s = 3$, steps (a) through (d) build a fully connected graph. As soon as there are more than s nodes already in the graph, new nodes will be connected to the graph by exactly s new edges (steps (e) and (f)).

C. Establishing a New Shared Key

Obviously, there is no need to establish additional shared keys for networks with up to $s + 1$ devices, since the corresponding network graph will always be fully connected. Thus, for the following discussion we only consider cases where $|V| > s + 1$.

We will first describe how two devices can establish a new shared key between them. In order to establish a l-bit key, a device randomly generates s l-bit shares $k_1, \ldots k_s$, and sends them over s device-disjoint paths (i.e. paths that do not share common devices) to the destination device (fig. 2). On each hop of a path, the key share is transmitted in an encrypted fashion using the existing appropriate shared key. The final key k is then calculated by $k = k_1 \oplus k_2 \oplus \ldots \oplus k_s$, where $\oplus$ is the bitwise XOR operation.

It should be clear that without having access to all key shares, an attacker does not stand a chance to recover the key. If it can be assured that the key shares are communicated over s node-disjoint paths of the network graph, an attacker

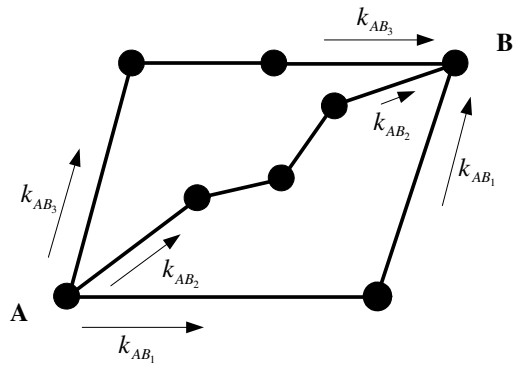

Fig. 2
LINK ESTABLISHMENT ($s = 3$)

will need to subvert at least s nodes (one on each path) to compromise the newly established key. In section VI, we will show that our construction does ensure this property. The s different paths can then be discovered using standard methods, e.g. through bounded depth-first search. Note, that due to the pairwise symmetric keys devices are authenticated against each other and therefore an single attacker device cannot force itself on multiple paths.

VI. PROOF OF CORRECTNESS

It will now be shown that the construction of the network graph described in section V-B will always result in s disjoint paths between any pair of nodes for all network graphs (V, E) with $|V| > s + 1$.

We use the following definition from graph theory:

***Definition** 1 (k-connected graph):* A graph $G = (V, E)$ is said to be *k-connected* if and only if for any set $W \subseteq V$ with $|W| < k$, the subgraph induced by $V - W$ is still connected.

***Theorem** 1 (Menger's Theorem [16]):* In a k-connected graph, there always exist k node-disjoint paths between any pair of non-neighboring nodes.

Proof: See, for instance, [17]. ■

***Theorem** 2:* Any graph $G = (V, E)$ as constructed using algorithm 1, with $|V| \geq s$ will be s-connected.

Proof: (By induction over $|V|$). For $|V| = s + 1$, the graph will be fully connected, so it follows trivially that the graph is also s-connected.

Now consider an already s-connected graph. We add a new node v using algorithm 1. So, we have s new edges into the existing graph. Since the original graph was already s-connected, after adding new edges, it will still be s-connected. And, in order to disconnect the new node, we would need to remove at least s other nodes. So s-connectivity cannot be violated by detaching the new node v either. ■

***Theorem** 3:* In any graph $G = (V, E)$ as constructed using algorithm 1, with $|V| \geq s + 1$, there will always be s node-disjoint paths between any pair of nodes.

Proof: Follows directly from theorems 1 and 2. ■

VII. PROPERTIES OF OUR APPROACH

Our approach to key distribution has the following properties:

- For the "skeleton" graph constructed in section V-B, i.e. the graph containing only the edges for keys that were established through physical contact, each device needs, on average, space to store $2s$ keys. This is easy to see, since each new device requires at most s keys (less for the first s nodes), each of which will be stored on two devices. With n devices, this leaves us with space for $2ns$ keys for the whole network, so if the keys are evenly distributed, each device will hold $2s$ keys. Since the network size does not influence the necessary storage space, the parameter s can be chosen according to the security requirements of the system, the susceptibility of the devices to tampering and the available memory on each device.
- The key distribution scheme remains safe as long as an attacker cannot subvert more than $s - 1$ devices at the same time. As long as this property holds, neither can the communication between two non-subverted devices be overheard by an attacker nor can an attacker fool a node about the origin of a message.
- Existing shared keys between untampered devices remain secure. Devices that forwarded key shares can immediately discard them afterwards.
- This scheme is still vulnerable against the second introduced type of attacker, the denial-of-service attacker. The straightforward solution to deal with this problem is the introduction of some redundancy into the network, i.e. in order to deal with a denial-of-service attacker that has compromised up to r nodes we need to establish a $(s + r)$-connected graph in the beginning. Thus, between any pair of nodes, there exist $(s + r)$ node-disjoint paths. If a node detects that a newly established key does not work, it chooses an alternative set of s paths to the other node and tries the key establishment again. A complete analysis of this mechanism is beyond the scope of this paper and will be discussed in future work.

Therefore, our approach provides a decentralized solution to the key-distribution problem based on symmetric keys that in the presence of only eavesdropping attackers can be guaranteed to work. There is no restriction on the addition of new devices and the memory required for key storage is, on average, constant for each device. Also, adding a new device only requires the user to touch it to max. s (or $s + r$) already existing devices, which is in line with our requirement regarding ease of use. What remains to be discussed is how a device can be removed from the network.

VIII. Controlled Removal of Devices

One problem remains to be solved: How can devices be removed from the network without destroying the property that the corresponding "skeleton" graph is s-connected? We assume that the removal of a device occurs as a controlled shutdown, i.e. a device has the time to announce its impending departure from the network and make all necessary arrangements. Algorithms 2 and 3 then describe the procedure for removing a device from the network.

Our solution is based on pretending that the device that is to be removed had never been there in the first place and replacing existing shared keys accordingly. If the device is still present when this procedure is performed, keys can be replaced automatically, using the procedure described in section V-C.

To see why our algorithms work, let the graph under consideration be $G = (V, E)$ with $V = \{v_1, v_2, \dots v_n\}$ where a node v_i was the i-th node added to the graph according to our construction. Let the node that is to be removed from the graph be v_j. We need to consider three cases:

1) If $|V| \leq s + 1$, removal is not an issue, since the resulting graph after removal of a node would remain fully connected.
2) $|V| > s + 1$ and $j > s$. Now, when v_j was added to the graph, s edges were established with already existing nodes. We call the set of these nodes $\mathcal{A} = \{v_{A1}, \dots, v_{As}\}$, where each index Ai is less than j (i.e. all nodes in $\mathcal{A}$ are older than v_j). Also there is a (possibly empty) set of nodes that were added to the graph after v_j and that, when they were introduced, established an edge with v_j. We call this set $\mathcal{B}$, and all nodes in $\mathcal{B}$ are newer than v_j (fig. 3). In the algorithms 2 & 3, the variable *FirstDevices* represents the set $\mathcal{A}$.
   If node v_j is removed, each node $w \in \mathcal{B}$ will be missing one of its original edges, possibly violating the *s-connected* property. In order to replace this edge, we need to establish a new link, and this link needs to be with a node that did already exist when w was added to the graph. Fortunately, since the set $\mathcal{A}$ has exactly s elements and w now shares at most $s - 1$ edges with older nodes of the graph, hence also at most $s-1$ edges with elements of $\mathcal{A}$, such a node can always be found in $\mathcal{A}$.
3) What now remains to be examined is the case where $|V| > s + 1$ and $j \leq s$, i.e. the node to be removed is among the first s nodes in the graph. In this case, we define $\mathcal{A} := \{v_i | 1 \leq i \leq s + 1 \wedge i \neq j\}$, $\mathcal{B}$ as in the previous case and proceed as before. After this is done, v_{s+2} will share links with all s nodes in $\mathcal{A}$, therefore $\mathcal{A} \cup \{v_{s+2}\}$ will be fully connected. Also, all $w \in \mathcal{B} - \mathcal{A}$ will have established a new link with a node in $\mathcal{A}$, so the *s-connected* property will be repaired.

Algorithm 2 Leaving the network

```
1: FirstDevices := first s known devices (set A);
2: MyDeviceList := the set of devices for which we have a
   shared key;
3: for all Device ∈ MyDeviceList do
4:    sendto(Device, LeavingIntention{FirstDevices});
5: end for
6: wait for all ACKs;
```

Algorithm 3 Action of a neighboring device

```
1: onReceiveLeavingIntention(Sender, DeviceList);
2: MyDeviceList := the set of devices for which we have a
   shared key;
3: Candidates := DeviceList − MyDeviceList;
4: if Candidates ≠ ∅ then
5:    Device := pick randomly one from Candidates;
6:    establish a new shared key with Device;
7:    update MyDeviceList by replacing Sender with Device;
8: else
9:    MyDeviceList := MyDeviceList − {Sender};
10: end if
11: sendto(Sender,"ACK");
```

It is necessary that every node is able to determine the elements of set $\mathcal{A}$ – even if the above algorithms are executed multiple times, i.e. the consecutive removal of multiple nodes. This can be achieved by maintaining a linear list of all neighbor-devices in the order of which they have become known to the device. When a device gets removed, two cases for processing this list have to be considered:

1) *Candidates* $\neq \emptyset$ (see algorithm 3), i.e. the node is an element of the set $\mathcal{B}$: The newly established key must **replace** the key which belongs to the leaving node.
2) *Candidates* = $\emptyset$, i.e. the node is element of the set $\mathcal{A}$: The key for the leaving device is deleted and the rest of the list is shifted.

As an example, consider figure 4. Device C is to be removed. It sends a message to all neighbors announcing its impending departure and includes the first s devices it has physically exchanged keys with (A, B, D). Nodes A, B and D already share keys with each other, so they will not need

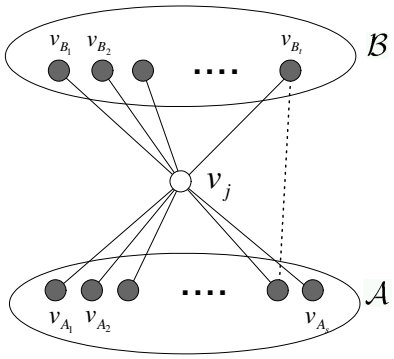

Fig. 3
Repairing Links

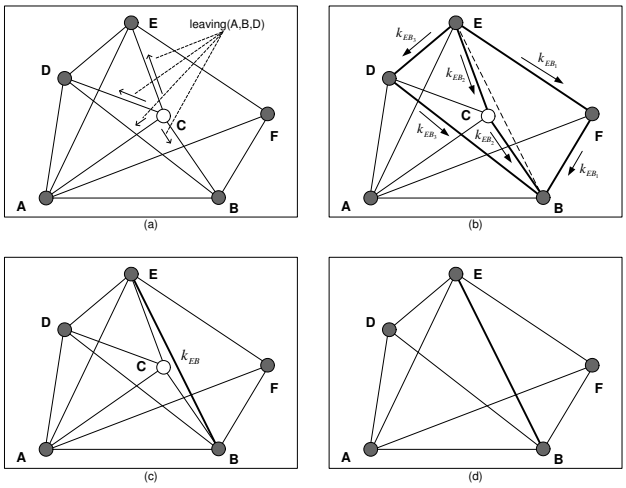

Fig. 4
REMOVING NODES FROM THE NETWORK GRAPH ($s = 3$)

to act. Node E, however, learns that it needs to establish a new key and the only option is to do so with B. A new key is established between E and B. After all other nodes have acknowledged, C can then leave the network.

IX. CONCLUSION AND FUTURE WORK

In this paper, we have presented an approach for key distribution in home automation networks based on symmetric cryptography. In contrast to the related work, we have proposed an algorithm for key distribution which guarantees the ability for an arbitrary pair of devices to exchange a key in a secure fashion, provided that the number of devices an attacker is able to subvert is not higher than the security level s of the network. We achieved this without prior knowledge of the maximum network size. Additionally, we do not require any pre-configuration of the devices.

The network can grow incrementally and also shrink if devices are removed in a controlled fashion. Adding a new device to the network only requires the user to physically connect it with s other devices to perform the key exchange, fulfilling our "ease of use" requirement.

The implementation of our approach is underway. Furthermore, we are working on mechanisms to discover the device-disjoint paths. Additional work includes a detailed analysis of node failures and denial-of-service attacks, algorithms for autonomous graph reconnection after node failures and key revocation schemes. In the near future, we plan to conduct performance evaluations on our approach.

ACKNOWLEDGMENTS

We thank Pedro Marron, Gregor Schiele, Christoph Ruffner, Jörg Hähner, Stefan Lewandowski and the anonymous reviewers for their valuable comments and help.

REFERENCES

[1] EIB - European Installation Bus. Webpage. [Online]. Available: http://www.eiba.com
[2] ELV - FS20 RF Remote Control System. Webpage. [Online]. Available: http://www.elv.de
[3] X10 Home Automation System. Webpage. [Online]. Available: http://www.x10.com
[4] R. Anderson and M. Kuhn, "Tamper resistance - a cautionary note," in *Proceedings of the Second Usenix Workshop on Electronic Commerce*, Nov. 1996, pp. 1–11.
[5] M. Brown and D. Cheung, "PGP in constrained wireless devices," in *Proceedings of the 9th USENIX Security Symposium*, 2000.
[6] J.-P. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in mobile ad hoc networks," in *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2001, pp. 146–155.
[7] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.
[8] N. Modadugu, D. Boneh, and M. Kim, "Generating RSA keys on a handheld using an untrusted server," in *Cryptographer's Track RSA Conference*, 2000.
[9] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Mobile Computing and Networking*, 2001, pp. 189–199.
[10] J. Kohl and C. Neuman, "The Kerberos network authentication service (v5)," IETF RFC 1510, Sept. 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1510.txt
[11] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure pebblenets," in *Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking and computing*. ACM Press, 2001, pp. 156–163.
[12] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy*, May 2003.
[13] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach," George Mason University, Tech. Rep. ISE-TR-03-01, Mar. 2003. [Online]. Available: http://www.ise.gmu.edu/techrep/2003/03_01.pdf
[14] L. Gong, "Increasing availability and security of an authentication service," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 657–662, 1993.
[15] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *7th International Workshop on Security Protocols*, ser. Lecture Notes in Computer Science, 1999, pp. 172–194.
[16] K. Menger, "Zur allgemeinen Kurventheorie," *Fund. Math.*, no. 10, pp. 96–115, 1927.
[17] F. Harary, *Graph Theory*. Perseus Publishing, 1995.