Secure Overlay for Service Centric Wireless Sensor Networks

Hans-Joachim Hof, Erik-Oliver Blaß, Martina Zitterbart

[hof | blass | zit@tm.uka.de]

Institut of Telematics, University of Karlsruhe

Abstract.

Sensor networks consist of a potentially huge number of very small and resource limited self-organizing devices. Those devices offer different services and use services provided by other sensor nodes. To give sensor nodes the possibility to offer services and to network-wide search for available services, some kind of lookup facility is needed. Several possibilities exist to realize service lookup in traditional networks and ad-hoc networks [ALM03, GOL99, GUT99, PRE02, SAL99, ZHU03]. In this paper we present Secure Content Addressable Networks Version 2 (SCANv2), a secure overlay focusing especially on wireless sensor networks. The paper describes how this secure overlay can be used among other things to offer lookup functionality in sensor networks. The design of the overlay focuses on secure service lookups. The overlay is part of the Karlsruhe Sensor Network Platform K-SNeP, a modular and flexible architecture for service centric sensor networks. Key areas of application of the architecture are gradually extendable service centric sensor networks where sensors and actuators jointly perform various user defined tasks, e.g. in the field of an office environment or health care.

1. Introduction

As computer miniaturisation continues and small devices become cheaper and cheaper, more and more computers are embedded into the user's environments to offer pervasive services which enhance the surrounding of the user with intelligence. Sensor networks are special occurrences of networks formed by those small devices. The nodes of those networks have low-power, are self-organising and have little computation capabilities. Such devices typically use "peanut CPUs" [STA02]. Sensor nodes usually offer services (for example the data which they acquire) and use services provided by other sensor nodes to construct even more complex services (for example a services which controls an actuator based on the output of a sensor). The goal of sensor networks is to benefit of synergy effects of a huge number of sensor nodes in a network which has a high node density.

Security issues are very important in sensor networks as sensors invade the personal environment of a user and at the same time are harder to recognise ("disappearing computer"). In many scenarios, like assisted living or health care, sensor networks have critical functions. Therefore if sensor networks should ever be widely used, it is crucial that they have high security standards.
Services are the building blocks of the functionality of any sensor network. Secure access to services and secure and robust service lookups are therefore vitally important for the security of the whole network.
Given the limited resources (low memory, little CPU power, low bandwidth) and other characteristics of sensor networks (maintenance-free operation, frequent node failures) it stands to reason that traditional security methods do not work satisfyingly in the context of sensor networks. Especially asymmetric cryptography seems computationally too complex for these nodes, even with latest techniques like elliptic curve algorithms [KOB94]. Therefore, security architectures for sensor networks benefit from focusing on symmetric cryptography which can be used efficiently even on peanut CPUs.
This paper focuses on secure and robust service lookups and describes a secure overlay for sensor networks that is used in the "Karlsruhe Sensor Network Platform" (K-SNeP) to realise among other things a distributed service directory. The security mechanisms of the overlay take into consideration the limited resources of sensor network nodes. Therefore, only symmetric cryptography is used on sensor nodes. The key idea behind the use of an overlay for wireless sensor networks is to create a simple distributed system that easily scales with its growth, is considerably failsafe, and does not require any centralised component during normal operation. Hence, a user can sequentially add nodes to the network without bothering to provide enough capacity. A special device, the so called Master Device, is used to bring sensor nodes into the network. This is the only device which may utilise asymmetric cryptography.
A typical scenario where K-SNeP can be advantageously deployed is an intelligent office environment with dozens or even hundreds of sensors and actuators that act jointly within a sensor network to accomplish various tasks. Another scenario is health care in a hospital where wireless sensors are used to replace wires and therefore make the work of doctors easier and the patients more mobile. In this scenario, sensors may also be build into a prosthesis to control correct movement of a patient. Sensors can even be used to record life signs of a patient over a long period of time even if the patient is not in the hospital. Assisted Living is yet another scenario which benefits from the "Karlsruhe Sensor Network Platform".

The paper is structured as follows: Chapter 2 presents Content Addressable Network which is the basis of the secure overlay presented in this paper. Chapter 3 shows the context in which the virtual overlay will be used: the Karlsruhe Sensor Network Platform. Chapter 4 presents the proposed secure overlay network: Secure Content Addressable Network Version 2 (SCANv2). This chapter also describes how SCANv2 can be used to build a distributed service directory. It also presents Clustered SCANv2, a solution for heterogeneous sensor networks which enables very low power devices to uses the distributed service directory. Chapter 5 summarises the paper and gives an outlook on future work.

2. Content Addressable Networks

Overlays are an emerging issue in wired networks. Filesharing tools like the open source project emule [EMU04] use those overlays for their services. Several overlays like Chord [STO01], Pastry [ROW01], Tapestry [ZHA01], Content Addressable Network [RAT01] and Kademlia [MAY02] are available. The Secure Content Addressable Network Version 2(SCANv2) presented in this paper is based on Content Addressable Network (CAN) which gets enhanced with security features. SCANv2 is an advancement of SCAN [HOF04]. CAN has been chosen because of its solid structure where neighbors in CAN space have a special relationship which can be easily secured. Another reason why we chose CAN is the low and constant memory overhead of CAN which is not the case with most of the other overlays. This chapter gives a short overview of CAN. Please refer to [RAT01] for more details.

CAN utilizes a d-dimensional Cartesian coordinate space on a d-torus. This virtual space will be called “CAN space” in the rest of this paper. The coordinate space is completely logical and has no relation to any physical coordinate system or to the structure of the network. CAN forms an overlay network which lies above the network layer and utilizes the communication abilities offered by it. Hence, CAN can only be functional as long as the underlying network layer is still usable. At any time, the entire CAN space is divided into zones which are administrated by the nodes participation in the CAN. Every node “owns” a distinct zone within the overall space hence such a node is called zone owner in the rest of this paper. New zones emerge from existing zones by a split considering an ordering of the dimensions. CAN realizes a distributed hash table. The virtual coordinate space is used to store *(key,value)* pairs as follows: *key* is mapped on a point *P* in CAN space using a hash function. The *(key,value)* pair is then stored on the zone owner of the zone within which *P* lies. To retrieve an entry corresponding to a key *K*, any node can apply the public hash function to map *K* on a point *P* and then retrieve the corresponding value from the zone owner of the zone in which *P* lies. The request is routed through the CAN space until it reaches the node which owns the zone including *P*. Routing is done by greedily forwarding a message to the neighbor which coordinates are closest to the destination. Each node keeps a list of neighbors that abut their own zone. This list is used for forwarding and acts as a routing table. Periodic update messages between neighbors help to recognize failed devices and abandoned zones. Please note that many different paths exist between two points in the CAN space. This means, that even if one or more of a node’s neighbors crash, a node would automatically route along the next best available path. If a node loses all its neighbors, it can do an expanding ring search to get connected to the CAN again. CAN describes some repair mechanisms for restoring a consistent state. Those are of no interest for this paper and will get careful attention in further work.
As mentioned earlier CAN space is partitioned among a number of nodes. If a new node decides to participate in CAN an existing zone is split into half and the joining device gets one of the two resulting zones. A uniform distribution of join points is eligible to maintain zones of equal size which is important because effective routing depends on a similar zone size of all zones. CAN offers some optimizations of routing and robustness. Routing can be improved using multiple CAN spaces (called multiple

realities) with different hash functions on each node. Reaching a point in CAN then translates to reaching this point in any reality. Robustness can be improved by zone overloading. This means that multiple nodes own one zone and all owners are permitted to issue answers for requests on this zone. If one owner fails, there are still other zone owners which may answer requests.

3. Karlsruhe Sensor Network Platform

The proposed secure overlay is embedded in the "Karlsruhe Sensor Network Platform" (K-SNeP) which is a flexible and general architecture for wireless sensor networks. K-SNeP was designed for a special occurrence of sensor networks: service-centric sensor network. Service-centric sensor networks focus on services and realization of services in a network. In service-centric sensor networks, data flows between sensors and actuators, one delivering data and the other executing control tasks. Both, data delivery and execution of tasks may be expressed as a service. Service-centric sensor networks are therefore contradictory to traditional data-centric sensor networks where the data flow is between a huge number of sensors and a small number (typically only one) of control stations. Data-centric sensor networks are typically used for environmental observation projects, for example Great Duck Island [GRE04].
Figure 1 gives an overview of the "Karlsruhe Sensor Network Platform". In the following, its modules will be described in detail.

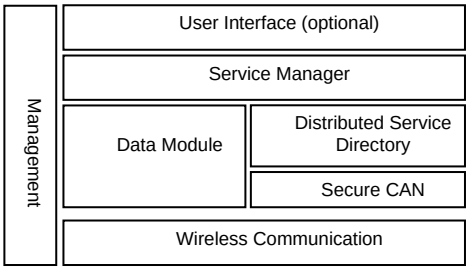

Figure 1: Karlsruhe Sensor Network Platform

3.1 Distributed Service Directory

As explained above, services are the building blocks of a sensor network. Complex services rely on the output of other services in the network and use services on actuators to execute control tasks. A flexible use of services can only be possible if a robust lookup functionality is present in the network. The Service Directory module implements this functionality. The interface and administrational functions of the Service Directory are independent of the concrete implementation. At the moment, K-SNeP may use one of two methods: In a totally decentralised scenario, the SCANv2 presented in this paper is used. In the presence of gateways on the other hand the functionality of the Distributed Service Directory will either be implemented on those gateways with help of services in the network to which the gateway connects or, in a

more general case, a clustered SCANv2 (see below) can be used where gateways are clusterheads. The service directory is used to store service records holding attributes of services which are available in the network. Attributes include, but are not limited to, address of service provider, address of data replication point, physical location of service, validity of service, quality category, needed input to provide service, output format etc. Attributes are service-dependant. A special query language allows the user to specify the attributes the sought-after service should have.
To deal with node failure it is necessary to demand constant refresh of service entries. Services can also have a limited lifetime after which their service records are removed from the service directory.

3.2 Secure Content Addressable Network

The Secure CAN (SCAN) module implements the secure overlay described in this paper. It realises a secure and robust distributed hash table. A Content Addressable Network (CAN) is used as base of the secure hash table and gets enhanced with security features. See chapter 2 for more information on CAN and chapter 4 for details on SCANv2.

3.3 Service Manager

The Service Manager pairs actuators and sensors to execute a user task. It supervises service execution. The pairing of actuators and sensors could be temporary or permanent, access could be simultaneous, competitive or exclusive. The Service Manager Module is also responsible to determine the sensor node on which a task should be executed. Services are described using a service description language and may be executed on extern nodes. The Service Manager defines and registers new complex services which are based on other services. The Service Manager is responsible for registering services in the Service Directory. It uses the Service Directory to find the needed services. As services may require results of other services as input, the Service Manager Module may need to do cascading lookups in the service directory. Please note that the Service Manager has only to do lookups at the time of pairing. Later on, the paired actuators and sensors do not need any more service lookups as they know each other.

3.4 Management

The Management Module realises all those functions which need access to all layers or provide information needed on all layers. The Management Module includes software updates and localisation. Software updates are vital for unattended sensor networks. They allow error correction, adaptation to unforeseen environmental conditions and recalibration of sensors. Location information can be retrieved using different methods on different layers. Nearly all layers need information about the position of the node.

3.5 Data Module

The Data Module is responsible for service-centric data aggregation, data replication and, on sensor nodes, for data acquisition.
The basic idea of service-centric data aggregation is that some requester defines an aggregation rule and finds an aggregation point, where the aggregation should take place. Aggregation is offered as a service and sensor nodes capable of executing aggregations register themselves and can be found by other nodes using the distributed service directory. The requesting sensor node defines the aggregation rule and a set of sensors which should be used for the aggregation. The aggregating node executes the aggregation rule repeatedly and registers in the service directory a new service which offers the aggregated data. Other nodes which need the same aggregated data may now use this service. If the aggregation rule is stored somewhere in the network, for example using the overlay presented in this paper, a node can request the sensor data and the aggregated values and verify the aggregation thus making it more secure as cheating may be easily noticed.
Data replication is used to cache frequently used data pieces. This can result in energy saving, if sensor values have a certain lifetime for which the sensor would acquire always the same or very similar data. Data replication may also be useful in data aggregation to deal with timing problems if sensor values are not acquired synchronously. Therefore, in the data module the data replication module is directly connected to the data aggregation module.
Data Acquisition is initiated by the Service Manager Module on sensors. Data acquisition may be initialised by another sensor node (pull) or may be executed based on an intern scheduler and published in regular periods to other sensor nodes (push)

3.6 User Interface

The User Interface enables the user to communicate with the sensor network. It gives her the possibility to issue commands to the sensor network and it provides a nice way to get results back. The User Interface is accountable for execution of commands and returns (aggregated) overall results of quests given to the sensor network. It may represent a gateway between the user's network (like the Internet) and an off-site sensor network. The User Interface may also be used for security issues. For example hardware of our testbed (called BlueEgg) has a build-in wheel to enter number sequences which can be used for authentication between two devices.

3.7 Wireless Communication

The proposed architecture requires some underlying wireless communication ability. Our testbed uses Bluetooth as communication layer. However, the architecture itself is independent of the actual used communication technology and can be implemented on most of the recently available sensor network platforms.

4. Secure Content Addressable Networks

The proposed protocol for Secure Content Addressable Networks is based on Content Addressable Networks presented in Chapter 2. However, CAN has some security flaws:

An obvious attack on a distributed hash table is to overtake a certain part of the hash table. In CAN, this can easily be done if an attacker can use an arbitrary join point. If an attacker would like to overtake a certain hash value h1 which lies in the zone (x1,y1,x2,y2) it would choose h1 as its join point. The owner of the zone would split its zone and hand over one part of the zone to the attacker (it is not necessarily the zone where h1 lies in). This can be done multiple times until the attacker gets the zone in which h1 lies.

Another attack may be performed if it is possible to claim to own an arbitrary zone of CAN. An attacker may than claim to own the whole CAN space or a huge part of it and therefore draw joining nodes into a fake network.

Communication between neighbors in CAN space (“one hop”) usually takes multiple hops on network layer. If neighbors want to communicate securely, for example to achieve a common purpose like isolating an attacker, neighbors need to have a symmetric key in common. Some information like the continuous update messages need not be secret but authenticated and integrity checked.

Those issues can be solved with an extension to CAN: Secure Content Addressable Networks (SCAN). The basic idea is to use a Master Device as trust anchor for the virtual overlay. The Master Device is not part of the sensor network itself. It is a device more powerful than the typically sensor node and has especially the ability to perform public key operations. However, the Master Device is designed to have only one hard-wired key and no other state information. As the Master Device has no state information which it collects during the lifetime of the sensor network, the Master Device can be easily replaced when lost or damaged. It may also be duplicated to give multiple users the possibility to bring new nodes into the network. The Master Device is only present when a new node joins the sensor network and is under total control of the user. The user authenticates the joining device by the use of a location limited channel [BAL02], e.g. physical contact or infrared. For an actual implementation of SCAN, the master device should be a small gizmo which the user can carry with him all the time. [MAX04] is perfect for our implementation. Given the properties of the Master Device, the first security flaw can be eliminated by letting the Master Device select the join point and ensuring, that no other join point can be used. To protect against the second security flaw, the Master Device issues zone certificates during the join of a node. Those certificates get checked later by the Master Device when another node joins. The Master Device is used to establish a common secret between neighbors.

The first version of Secure Content Addressable Network was presented in [HOF04]. However, this version has some problems:

First of all, there is no way for the Master Device to notice attacks. More specific, there is no feedback during the join of a new device.
Secondly, the Master Device must be able to order other nodes to act upon a possible attack.
Thirdly, in the first SCAN proposal an attacker had the possibility to use old and no longer valid zone certificates. This is especially a problem when the hardware is not tamper-proof because in this case, it is likely that it would be possible to reconstruct an old zone certificate from the memory of an attacked sensor nodes.
Fourthly, in the first version of SCAN , symmetric keys are constructed on sensor nodes. However, those nodes are likely to have no good random number generator and the keys may therefore be insecure.

To deal with all those problems we present in the following a second version of the Secure Content Addressable Network protocol called SCANv2.

4.1 Assumptions on the attacker

Sensor networks typically cover environments where an attacker has physical access to sensor nodes. The sensor nodes are naturally not supervised in an public environment. Sensor networks should consist of a large number of very cheap devices. Therefore we do not expect sensors to be tamper-proof.

Considering those properties of a sensor network, an attacker can remove any sensor from the network and impersonate any sensor in the network if he has physical access. An attacker can also clone sensor nodes and add fake nodes to the network at his will. However, he can not add nodes to SCANv2 as the join process includes an interaction with the Master Device.

An attacker may eavesdrop any local communication.

There may be special devices which can not be tampered with. In our case, the Master Device is considered tamper-proof and as it is under surveillance of the user (it may be a key fob which the user carries with him all the time), chances are low that an attacker gets access to this device.

An attacker can not easily distinguish the physical position of a node knowing only the administrated zone in the SCANv2 as the structure of the overlay is unrelated of the actual network structure.

We expect it impossible to attack a sensor node before it is integrated in the sensor network by the user. The sensor node may be sealed by the vendor and the vendor guarantees that the sensor has not been infiltrated at the point of sealing.

4.2 Secure Construction and Preservation of Structure in SCANv2

The description of the proposed protocol uses the following syntax:

Channel | A -> B: Message

Channel denotes the channel a message is send in. In the present case, this can be the location limited channel, for example physical contact (*PHY*), the main communication channel like Bluetooth Scatternets (*MC*) or the CAN overlay (*CAN*). *A* denotes the sender and *B* denotes the receiver of *Message*. In the case under consideration, sender and receiver can be Master Device (*MD*), Joining Device (*JD*), owner of the zone *JD* wants to join into (*ZO*) and neighbors of *ZO* (N_i).

The Protocol consists of ten steps:

1.) PHY | MD -> JD: JP, E_{SK}*(JP,"temp")*

First, the Master Device sends *JD* over the Location Limited Channel (physical contact in this case) the join point (*JP*) at which JD will join the CAN. This point is selected randomly by the Master Device. Random selection is very important to achieve a equal distribution of zone sizes in CAN to ensure that routing in the CAN is ideal. It also ensures that it is not possible to overtake a specific part of the distributed hash table by a specially chosen join point. The Master Device also sends the join point encrypted with its super key (*SK*), which is the private key of the Master Device. The encrypted join point is used as a shared symmetric key between the node and the Master Device. It will be denoted with *k* in the further description of the protocol. The string "temp" indicates the temporary character of the key. It is important to always ensure that the join point of a device lies in its owned zone and there are cases where it is necessary for a joining device to get another join point in a later step to ensure this characteristic.

2.) CAN | MD -> ZO: "who is responsible for JP", Address

Next, the Master Device sends a message into the CAN to find out, who owns the zone *JP* lies in. This message is not broadcasted but send in CAN space to the address *JP*. The message will therefore reach the zone owner which answers immediately. The message also includes the network layer address of the Master Device to enable the receiver to communicate with the Master Device outside the CAN on network layer. This is done for performance reasons as communication on network layer is more efficient than communication in CAN space. As stated earlier, it is not necessary for the Master Device to have the ability to communication with the network because it could otherwise use the communication abilities of the trusted joining device over the location limited channel. However, for simplification of the protocol, we assume the Master Device to be able to communication with the CAN in our further protocol description.

3.) MC | ZO -> MD: JP_{ZO},E_k(certificate, time, NeighborJPList)

The zone owner answers the request of the Master Device with its join point and the certificate of its zone (see below), encrypted with the key shared between the Master Device and the zone owner. The encrypted message also contains a timestamp to prevent reply attacks. The Master Device derives *k* using its private key as described earlier. It checks the zone owners certificate. The size of the owned zone is included in the certificate. The message from the Zone Owner also includes a list of join points of the neighbors of the zone owner in the formate $((n_1,JP_1),(n_2,JP_2),..., (n_d,JP_d))$ where n_i is the network layer address of a neighbor and JP_i the join point of the corresponding node. This is a major chance in SCANv2 as the knowledge of the neighbors join points and network layer addresses enable the Master Device to communicate with the neighbors of the zone owner. The join points are used to derive the keys the Master Device has in common with the neighbors. A feedback mechanism can now be constructed to make the Master Device aware of possible attacks during a join operation and the Master Device has a way to give orders to the neighbors of the zone owner.

4.) PHY | MD -> JD: JP, E_{SK}(JP,"perm")

The Master Device evaluates if the joining node needs a new join point taking in consideration the join point of the zone owner and the future split line in CAN. The goal is to ensure that the join point of a zone owner always lies in its zone. See chapter 2 for a description of zone splits. In all cases, the joining device gets a new symmetric key shared with the Master Device, which is constructed in the same way as described earlier. This time, key and join point are flagged as permanent. For any further interaction of the joining node with the Master Device, only the permanent key and therefore only the permanent join point is valid. From now on the sensor node uses the symmetric key for authentication and secure communication with the master device.

5.) PHY | MD -> JD: $key_{JD,ZO}$,neighborKeyList,, E_{ZO}("JP,JD", $key_{JD,ZO}$, $certificate_{ZO}$, E_{JD}($certificate_{JD}$), neighborInformationList)

In the next step, the Joining Device gets a symmetric key ($key_{JD,ZO}$) for secure communication with the zone owner and also a neighbor key list in the format (neighbor address, symmetric key, join point). The symmetric keys are used for secure communication between the new neighbors after the zone split. Unlike the first version of SCAN, in SCANv2 the symmetric keys are constructed on the master device because it is not ensured that the neighbors all have a good random number generator. The join points are stored in the neighbor table of the Joining Device. The Master Device also hands out a "ticket" to *JD* which enables it to initiate the zone split and to get its new zone from *ZO*. The ticket is encrypted with *ZO*'s symmetric key and includes address (*JD*) and join point of *JD*, a symmetric key for communication between *JD* and *ZO*, a certificate for *ZO*'s zone after the split operation ($certificate_{ZO}$) and a certificate for *JD*'s new zone ($certificate_{JD}$) encrypted with the symmetric key shared between the Master Device and the Joining Device.

The encrypted part includes some information for the neighbors of the Zone Owner which are affected by the zone split. The neighbor information list consists of several tuples in the format (zone size of zone owners zone, symmetric key for communication with Joining Device, symmetric group key for communication with all neighbors of the zone owner, Join Point of Joining Device). These tuples are encrypted for each neighbor with the symmetric key the Master Device has in common with each neighbor. Those keys can be derived by the Master Device with the help of the neighbors join points which have been transmitted in step 3 of the protocol (see there for details).

6.) MC | JD -> ZO: E_{ZO}*(“JP,JD”,* $key_{JD,ZO}$ *,* $certificate_{ZO}$ *,* E_{JD}*(*$certificate_{JD}$*), neighborInformationList)*

JD hands the ticket to *ZO*. *ZO* starts the zone split.

7.) MC | ZO -> JD: E_{key}*(data of hash table)*

ZO starts the zone split with the transmission of data stored in the part of the distributed hash table, which will belongs to *JD*. Transfer of data is encrypted with $key_{JD,ZO}$ which *JD* and *ZO* got handed out by the Master Device. This ensures, that no intermediate node is able to alter the data stored in *ZO*’s former hash table. Encryption is important to ensure integrity of the distributed hash table. All received data is acknowledged by *JD*. Those messages are not included in our protocol discussion for simplification.

8.) MC | ZO -> JD: E_{key}*(*E_{JD}*(*$certificate_{JD}$*))*

When the data transfer successfully ended, *ZO* hands out *JD*’s zone certificate. With the certificate being handed over at the end of the data transfer, *ZO* could test some values it formerly stored. This does not prevent the joining device to drop all data anyway, but it makes cheating more difficult. From the moment of certificate handover forth, *ZO* is no longer responsible for the half of its former zone. *ZO* immediately and thoroughly destroys its old zone certificate and stores the new one which was included in the last message from the Master Device. Unlike in version 1 of Secure Content Addressable Networks, in SCANv2 it is not that vital to thoroughly destroy old certificates because now the neighbors double check the zone size and report to the Master Device. Therefore, even if an attacker could reconstruct a deleted certificate, it has no value to him.

9.) MC | ZO -> N_i *:* E_{Ni}*(E(zone size,* $key_{JD,Ni}$*),JD_Address)*

As the zone split is now official in effect, *ZO* notifies all affected neighbors about the split and transfers them the neighborhood information, which was encrypted by the Master Device and can be only decrypted by the neighbors. This message includes the claimed zone size of the zone certificate of the zone owner and a symmetric key shared between the neighbor and the Joining Device. The message from the zone owner also includes the network layer address of the Joining Device. The message is

encrypted with the symmetric key *ZO* shares with each of its neighbors. The neighbors check if the zone size is correct from their local view.

10.) MC | N_i -> MD: JP, E_{Ni}(ACK,certificate) or JP, E_{Ni}(NACK, certificate)

If the zone size is correct from the local view of a neighbor, it sends an acknowledge to the master device. If the zone size is not correct, the neighbor sends a negative acknowledge. In all cases, the zone certificate of the neighbor is included and gets checked by the master device. The master device now has the prove that something is wrong and either the zone owner or the neighbor uses an old certificate. It is the challenge of the master device to resolve this issue. This feedback mechanism is new to SCANv2 and very helpful. The MD has several possibilities to find out who is cheating:
All affected neighbors either send ACK or NACK. A majority vote can be used to find out who is cheating. The master device could also query neighbors of the neighbor to verify the neighbors zone certificate. This however results in a high overhead. The age of a certificate can be used to estimate trust in combination with the knowledge of the ordering of dimensions during a zone split and the usual proportion of zone size (for example in the two dimensional case a zone has always a proportion between x and y out of the set {1:1, 1:2 and 2:1}).
To resolve the problem, the MD randomly selects one neighbor to become new zone owner (ideally the one which owned the zone earlier) and declares this node to be the new zone owner. The MD issues a new certificate and notifies all neighbors about the change. Additionally the MD can order the neighbors to add the zone owner to their ignore list.

4.3 Node Failure

Neighbors notice node failure when the constant update messages between neighbors are missing. They inform the other neighbors about this using the symmetric group key shared among them. They then start an election process and declare one node new zone owner. The zone owner gets the correct election receipted by all neighbors, encrypted with the key each neighbor has in common with the Master Device. This receipt also includes the (encrypted) zone certificate of the neighbor. At the next join which involves the zone owner (either as neighbor or as zone owner whose zone gets split) the Master Device checks the receipts and issues a new certificate. This process again needs communication with the neighbors of the zone owner to ensure that the claimed zone size is the actual zone size.

4.4 Usage of SCANv2

In the beginning of our work, the Secure Content Addressable Network Version 2 was mainly used to realise a secure distributed service directory. As stated earlier, K-SNeP is an architecture for service-centric sensor networks. A basic lookup-functionality is therefore crucial for the network to ensure, that service executors are able to find

matching sensors and actuators. However, SCANv2 may also be used for other purposes.
The above described service directory can be extended to act as a security anchor for secure services. For example, a commitment to a public key may be included in a service description, similar to self-certifying path names [MAZ99]. SCANv2 may also be used by the data module: in the process of data aggregation, aggregation rules and data descriptions may be stored in SCANv2 to be accessible for everybody who wants to get aggregated data making therefore the data aggregation transparent and verifiable.
However, for all usage of SCANv2, it is important to remember that one hop in the SCANv2 space can be multiple hops on network layer. Communication in SCANv2 is therefore energy expensive and should only be used in rare cases like in the cases above, where a node typically needs only one time contact to SCANv2 and this contact is followed by a long period in which the returned data is used.

4.5 Clustered SCANv2

Sensor networks are usually very heterogeneous in performance of the sensor nodes. Some sensors may not be able to perform all the task related with SCANv2. Because SCANv2 as basis of the distributed service directory is vital for service provisioning and usage of services, even those devices need to have a way to get information out of SCANv2. We propose to cluster those low-power devices around a more powerful sensor node (called cluster head) which provides all the information the cluster member needs. We expect that there are enough powerful sensor nodes regarding the complex tasks some sensors fulfil. Communication in clusters is usually one-hop communication. A hash-chain or a symmetric key is used to secure communication between cluster head and cluster members. Security is also set up by the Master Device. It may for example issue an up-to-date hash value to the joining node or hand it a symmetric key for communication with the cluster head. The Master Device could either hand out a personal symmetric key or a cluster key which is used in the whole cluster. A special case of a clustered SCANv2 is a scenarios where gateways are present which connect to other networks. There, gateways are cluster heads and the only SCANv2 members. Another special case is a clustered SCANv2 where the task of cluster head and therefore SCANv2 member changes from time to time between the mass of cluster members. This may be done for energy loss balancing between a group of sensors.

4.6 Redundancy and Load Balancing

The distributed service directory uses SCANv2 to store service records under the hash of the name of the service. This can be a problem if there is a huge amount of sensors which offer the same service because in this case, all the service records are stored in the same zone (on the same node). This uses resources of just one node and makes this node to a single point of failure and therefore a good attack target. The original CAN paper suggests to use “zone overloading” or “multiple hash functions”. With

zone overloading active, one zone may be owned by more than one node. Multiple hash functions can be used to calculate more than one hash value under which service records are stored. Both methods add redundancy but they distribute exactly the same amount of data on a higher number of nodes. However, communication will be less on the individual nodes. As communication consumes the main part of the energy reserves of a sensor node, both methods are used in K-SNeP. To achieve a better distribution of hash values for a service, each service may use an arbitrary but well-known parameter which is added to the service name before hashing. Typically, a discreet and often-used parameter is used because this makes searches more efficient. For example if a sensor network is used in an office environment and there is a special service which is used in context of a room, the room number may be appended to the service name thus distributing load and making searches more efficient as no compare operation is needed in the distributed service directory module on the zone owner.

5. Summary and Outlook

This paper presented a secure overlay for wireless sensor networks and how it can be used to realise a distributed service directory. The overlay is embedded in the Karlsruhe Sensor Network Platform for which this paper gave a work-in-progress status. The paper showed how common secrets are distributed to SCANv2 neighbors during the join of a new device and how zone certificates are used to prevent a node from claiming ownership of an arbitrary zone size. No asymmetric cryptography is used on any sensor network node. The paper presented a feedback mechanism to deal with problems during the join. It also presented how a group key is established between neighbors of a zone.
CAN in its basic implementation does not take into consideration the location of devices. This means, that eventually, communication of two physical neighbors takes plenty of hops in CAN. There are some enhancements of the basic proposal which we plan to integrate into our protocol.
Further research is needed to use SCANv2 and the service directory as trust anchors for secure services.

References

ALM03 Florina Almenárez, Celeste Campo: “SPDP: A Secure Service Discovery Pro-tocol for Ad-hoc Networks”, 9th Open European Summer School and I-FIP Workshop on Next Generation Networks, Budapest, Ungarn, 2003

BAL02 Dirk Balfanz, D. K. Smetters, Paul Stewart and H. Chi Wong: “Talking To Strangers: Authentication in Ad-Hoc Wireless Networks“, Symposium on Network and Distributed Systems Security (NDSS'02), Xerox Palo Alto Research Center, Palo Alto, USA, 2002

EMU04 http://www.emule-project.net/ , accessed on 09.03.2004

GRE04 http://www.greatduckisland.net/, accessed on 09.03.2004

GOL99 Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright: “Simple service discovery Protocol/1.0 operationg without an arbiter” , Internet Draft, IETF, 1999

GUT99 E. Guttman, C. Perkins, J. Veizades, M. Day: „Service Location Protocol, Version 2“, IETF RFC2608, 1999

HOF04 Hans-Joachim Hof, Erik-Oliver Blaß, Thomas Furhmann, Martina Zitterbart: “Design of a Secure Distributed Service Directory for Wireless Sensornetworks”, First European Workshop on Wireless Sensor Networks (EWSN), Berlin, 2004

KOB94 Neal Koblitz, “A course in number theory and cryptography, 2nd edition”, Springer Verlag, Berlin, 1994

MAX04 Maxime: “Java-Powered Cryptographic iButton”, http://www.ibutton.com/ ibuttons/java.html, accessed on 09.03.2004

MAY02 Petar Maymounkov and David Mazières, ”Kademlia: A peer-to-peer information system based on the XOR metric”, In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS ’02), Cambridge, March 2002

MAZ99 David Mazières, Michael Kaminsky, M. Frans Kaashoek and Emmett Witchel: “Separating key management from file system security”, in 17th Symposium on Operating Systems Principles (SOSP'99), Kiawah Island, 1999

PRE02 Stephan Preuß: “JESA Service Discovery Protocol: Efficient Service Discove-ry in Ad-Hoc Networks”, 2nd International IFIP-TC6 Networking Conference, Pisa, Italien, 2002

RAT01 Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker, “A Scalable Content-Addressable Network“, In Proceedings of ACM SIGCOMM 2001, August 2001

ROW01 Antony Rowstron and Peter Druschel: “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”, 18th Conference on Distributed Systems Platforms, Heidelberg, Germany, 2001

SAL99 The Salutation Consortium: “Salutation Architecture Specification Version 2.0c”, http://www.salutation.org/spec/Sa20e1a21.pdf, 1999

STA02 Frank Stajano, “Security for ubiquitous computing“, John Wiley & Sons, West Sussex, England, 2002

STO01 Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan: ”Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”, Technical Report TR-819, Massachusetts Institute of Technology, Cambridge, USA, March 2001

ZHA01 B.Y. Zhao, K.D. Kubiatowicz and A.D. Joseph: „Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing”, Technical Report UCB//CSD-01-1141, Computer Science Division, U. C. Berkeley, Berkeley, USA, April 2001

ZHU03 F. Zhu, M. Mutka, L. Ni: “Splendor: A secure, private and location-aware service discovery protocol supporting mobile services“, 1st IEEE International Conference on Pervasive Computing and Communications, 2003