A Security Architecture for Personal Networks

Assed Jehangir, Sonia M. Heemstra de Groot
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, The Netherlands
{jehangira, heemstra} @cs.utwente.nl

***Abstract*—Personal Network (PN) is a new concept utilizing pervasive computing to meet the needs of the user. As PNs edge closer towards reality, security becomes an important concern since any vulnerability in the system will limit its practical use. In this paper we introduce a security architecture designed for PNs. Our aim is to use secure but lightweight mechanisms suitable for resource constrained devices and wireless communication. We support pair-wise keys for secure cluster formation and use group keys for securing intra-cluster communication. In order to analyze the performance of our proposed mechanisms, we carry out simulations using ns-2. The results show that our mechanisms have a low overhead in terms of delay and energy consumption.**

Keywords-Personal Networks; secure communication; security agent, group key; key management

I. INTRODUCTION

The goal of our security architecture is to provide users of Personal Networks (PNs) [1] with a reliable communication platform to access their services. A PN comprises a core consisting of a PAN (Personal Area Network) which can be extended on demand to include other devices belonging to the user, both in his vicinity and those at remote locations such as the home or office. This transparent extension of the PAN will physically be made via infrastructure-based networks such as an organizations intranet, other ad-hoc networks, etc. Devices belonging to the PN can have one or more wireless interfaces such as Bluetooth, IEEE 802.11, UWB and ZigBee. These devices are expected to be mobile and their membership status as well as their physical location within the PN can change at any time. Fig. 1 outlines the PN architecture developed in the QoS for PN@Home project [2].

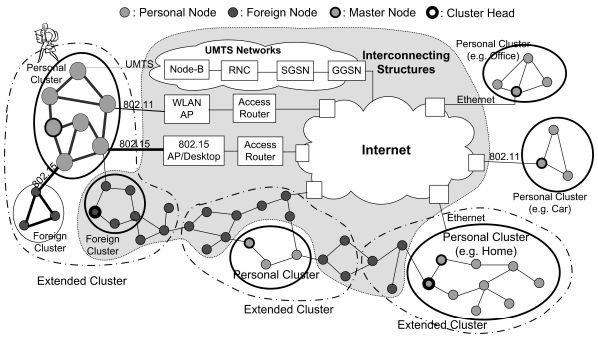

Figure 1. Clusters in a Personal Network

Providing security for PN devices is a major challenge since the majority of current security algorithms were designed for powerful workstations and are not practical for resource constrained devices. As a result, often for the sake of feasibility and efficiency, security is sacrificed. We believe that as technology advances and devices become more ubiquitous, strong security is necessary for a viable system. As discussed in [5], [6] and [7] the limited computational and energy resources of many PN devices imply that any proposed security solution must be simple and lightweight in order that it does not create a performance bottleneck of its own. Therefore our system is based on symmetric cryptographic primitives where public key cryptography is optional and only used between two parties when mutually agreed.

Since energy is the scarcest resource in our system, our security mechanisms must be frugal in their power consumption. We would like to minimize the security overhead in data packets and also restrict key management activities in order to conserve power. Fortunately since devices from a particular PN share the same owner, we believe that it is sufficient for PN devices to demonstrate group membership of a cluster rather than their individual identity. Such a restriction improves on system efficiency by using a shared group key (henceforth called the cluster key) to verify cluster membership instead of as many keys as members in the cluster. It reduces the overhead associated with key management such as the amount of processing required, the number of messages exchanged, authentication delay and storage space.

As illustrated in Fig. 1, personal devices initially organize themselves in the form of clusters. A cluster is defined as a collection of personal devices which can communicate amongst each other without using any non-personal devices. In order to operate as a cluster some devices perform additional organizational tasks. Fig. 1 shows two roles; that of a master node and a cluster head [4]. In this paper we define a new role, that of a **security agent** (Section III.A). Our proposed architecture requires each cluster to have one device functioning as a security agent; however there is no restriction to any other role that such a device may serve.

This remainder of this paper is organized as follows: Section II describes related work; Section III introduces some basic architectural concepts and Section IV explains the security requirements of our system. Section V describes how devices form clusters while Section VI explains key management, including device authentication/eviction and key

updates. Section VII analyses the performance of our proposed mechanisms in ns-2 and Section VIII concludes the paper.

II. Related Work

The only PN specific security architecture we know of is the work done in MAGNET [15] [16]. For a device to join a cluster it needs to have a long-term security association with any one neighboring clustered device. Cluster merge and splits do not post a challenge as there is no centralized management such as a security agent. The main weakness of the MAGNET approach is its large overhead. Since traffic sent between neighbors is encrypted using pair-wise keys, the system uses hop by hop encryption for both unicast and broadcast traffic. A large number of messages must be exchanged for a device to establish the required security associations with all its neighbors, particularly during high mobility. Our approach of using a shared key reduces the overhead of key management significantly. Additionally, messages no longer need to be re-encrypted at each hop.

The SPINS [5] set of protocols are designed for sensor networks where the network topology is rather different from that in a PN. Networks form around a base station where sensor nodes establish a routing tree, with the base station at the root. Devices establish a master key with the base station upon introduction to the network, and use that key to derive two new keys for protecting unicast traffic between the base station and themselves. Routing is much simpler since devices are only required to have a path towards the base station. A secure routing tree is created (for traffic to the base station) by having the base station send routing beacons protected using µTesla. Lastly, since battery replacement is designed to delete all the keys, there is no key management.

TinySec [6] is the first implemented link layer security protocol for sensor networks. As with our proposed mechanisms, it uses a shared symmetric key which is used by senders to first encrypt the data and then apply the MAC. The receiver uses the MAC to verify that the packet was not modified in transit. The TinySec protocol is not a complete architecture in the sense that it does not cover aspects such as key management and device discovery etc. However, the TinySec protocol has been well analyzed (and designed) in the context of sensor devices, especially related to its energy and communication overhead.

III. Architectural Concepts

Our approach is based on establishing a line of defense between the PN and the rest of the world by distinguishing between cluster members (trusted) and non-members (un-trusted). The semi-independent nature of clusters coupled with the fact that inter-cluster connectivity is not always guaranteed (even when possible it is infeasible to maintain it at all times for key synchronizations etc.) motivates our separation of trust at the cluster level.

Although we use group authentication for increasing efficiency, we appreciate that key compromise can have serious consequences since there is no easy way to uniquely identify the attacker. Unfortunately there are no ideal solutions for source authentication in group communication, especially in the case of multiple senders. One possibility is to use public key cryptography, but it is costly to generate and verify digital signatures on every packet. There exist schemes with the source authenticity allowed by public key cryptography but without the performance penalty [8] [9] [10], however they have their drawbacks.

A. Security Agents

The security agent authenticates new devices that join the cluster and initiates the periodic cluster key updates. It also periodically broadcasts cluster advertisements (CAs) which are used by other devices to discover the cluster. Additionally, the security agent is able to evict short-term members on demand and is responsible for setting the cluster policy which lets devices joining the cluster know about various cluster parameters like the frequency of cluster advertisements and key updates, and the length of certain timers etc. We decided to use a centralized mechanism for cluster authentication to ease management, reduce the number of security associations that need to be established and to limit the vulnerability of the system to compromised devices.

In terms of security agent functionality, we also define two new classifications of devices. Some devices have the capabilities to function as security agents and others do not. When not connected to other devices, a Security Agent Capable (SAC) device will function as a security agent and is thus considered a special form of a cluster which only contains itself. Security Agent Incapable (SAI) devices, unable to act as security agents, do not constitute a cluster when alone and need to join existing clusters. We expect SAI devices to be less sophisticated and typically not useful by themselves. They are designed to be used in conjunction with other smarter devices, when networked together as a cluster.

B. Cluster Key

The cluster key introduced previously is used to guard against unauthorized access that can degrade the quality of service for PN users. The cluster key is randomly generated for each time period, and given to new members by the security agent after a successful authentication. Once devices receive the cluster key they are able to take part in intra-cluster communication. The cluster key is periodically refreshed by the security agent and distributed to all existing cluster members.

Devices append a MAC (message authentication code) calculated using the cluster key to all cluster traffic that they generate. Consequently any subsequent device receiving the traffic can verify that it was generated by a trusted device and not modified in transit by any un-trusted device. Any packet with a source address of the cluster that does not have a valid MAC is dropped. The security provided to cluster traffic by the communication infrastructure is therefore group message integrity and authentication.

As appending the MAC increases the original packet size and thus the communication overhear, it should not be too large. Conventional security protocols use overly conservative security parameters, having MAC size of 8 or 16 bytes [6]. A larger MAC reduces the chance of an adversary blindly

guessing the appropriate code but correspondingly costs more to transmit. Reference [6] validates using a 4 byte MAC and explains how it is not detrimental in the context of low bandwidth links.

Earlier we pointed out the performance advantages of having devices authenticate themselves as part of the trusted group. However it becomes more difficult to identify any malicious behavior from inside the cluster as the attacker can alter his source address to that of another cluster device and still authenticate as part of the cluster. Even though such malicious behavior (typically due to device compromise) will be rare, it is something that can conceivably happen. In order to keep our mechanisms lightweight we do not attempt to identify malicious behavior of trusted devices. However our design allows the user to blacklist any suspect devices from the PN.

IV. SECURE COMMUNICATION

In this section we look at the security properties of our system. In order to protect the communication infrastructure and secure applications, devices require packet authentication. Therefore they will not accept invalid messages injected by an adversary.

A. Message Integrity and Authentication

Since every new cluster key is encrypted using the existing cluster key (Section VI.A), we limit further direct use of the cluster key in order to minimize its exposure to attack. If the cluster key were to get compromised then there is no forward secrecy because the attacker can use it to decrypt further re-key packets. Therefore the key used to generate the MAC (K_{mac}) is derived from the cluster key using a globally known one way hash function. This way we maintain forward secrecy because even if K_{mac} is compromised it can not be used to derive the existing or the future cluster keys. Similarly, backward secrecy is also guaranteed since only the security of the current session is compromised.

B. Encryption

As the aim of our security architecture is the dependability of the communication infrastructure and not confidentiality, messages are not automatically encrypted. Moreover, because encryption/decryption consumes power and increases latency, it is difficult to justify confidentiality as a basic requirement for all traffic. Most applications that require confidentiality already encrypt their traffic end-to-end, so duplicating the same functionality at the lower layers is not efficient.

If necessary, data confidentiality *can* be supported at the link layer by encrypting sensitive messages with an encryption key (K_{encr}). Similarly to K_{mac}, the encryption key is also derived from the cluster key using a globally known one-way transformation. This is also done so as to limit as much as possible the direct use of the cluster key.

V. CLUSTERING

Clustering, the process by which all PN devices within each others transmission range connect to form a cluster, is an integral part of a PN. Therefore one aim of all devices is to discover others around them that belong to their own PN. In principle, this is done by listening for cluster advertisements (CAs).

As mentioned before, SAI devices are envisioned to operate only as part of clusters and to be less sophisticated (a good example is that of a sensor). These devices can be left powered up for extended periods either purposely or mistakenly by the owner of the PN. We do not want such otherwise idle devices to spend precious energy continuously advertising themselves to non-existing neighbors. Clusters on the other hand, as shown by their interconnected state, are more active in nature. Therefore, only clusters are allowed to advertise, while un-clustered devices periodically wake up to listen for such advertisements. When an un-clustered device receives a cluster advertisement from a cluster belonging to its own PN, it will attempt to authenticate itself to, and join that cluster

Cluster advertisements are periodically generated and broadcasted by the security agent. Other cluster members re-broadcast non-duplicate advertisements, in effect "propagating" them to the edged of the cluster like a ripple on a pond. The periodicity of cluster advertisements and the decision on which devices take part in propagating these advertisements depends on the cluster policy at the security agent. Each member receives a copy of this policy when it joins the cluster.

A. PN and Cluster Address

We imagine the world to be full of wireless devices, belonging to a multitude of users. Therefore we need an efficient mechanism for PN devices to distinguish between those belonging to their own PN from all the rest. This is because we do not want devices from one user to continuously try to connect to others of different users and failing, wasting precious energy in the process. Therefore we assume the existence of a PN address, calculated *randomly* over a sufficiently large space so as to reduce the possibility of collisions. Devices learn of their PN address during the initial imprinting phase. Although the exact type of this address is beyond the scope of this document, it is conceivable that it could belong to either the IP domain, or a new layer between the IP and the MAC layers. Devices check the PN addresses of any cluster advertisement they receive, and only attempt to authenticate with clusters with similar PN addresses.

Additionally, we also assume the existence of a cluster address so cluster members can distinguish between traffic from inside and outside the cluster. The cluster address is derived from the physical address of security agent, and should therefore be unique. Cluster advertisements include both the PN address and the cluster address.

B. Cluster Dynamics

Cluster advertisements do more than just advertise the existence of a cluster to non-members; they also let cluster members know that their cluster is "alive". A cluster that is alive has a functioning security agent and can therefore grow by adding new members. Conversely a zombie cluster is one that has lost its security agent (but has a valid cluster key). Devices belonging to a zombie cluster can still communicate

securely with each other, but the cluster can not grow because there is no security agent to authenticate new members.

Devices believe they have lost connectivity with the security agent (and that they are part of a zombie cluster) when they miss "i" number of sequential cluster advertisements. The value of "i" is set by the security agent of the cluster and given to each member (as a part of the cluster policy) when it joins the cluster. Devices that believe they are part of a zombie cluster take steps to reorganize themselves. If possible, devices try to join other clusters belonging to their PN. SAC devices *may* also choose to leave a zombie cluster to create one of their own. Zombie clusters can only be resuscitated by the return and the resulting cluster advertisement generated by the original security agent. Since each security agent stores some state information about its cluster, it is not possible to recover from the loss of a security agent by re-election, only by creating a new cluster.

In the absence of a security agent, as there is no one to update the cluster key of the zombie cluster, it will eventually expire. If devices have not been able to join new clusters in the meantime, when this happens, they can no longer communicate amongst each other. SAI devices enter the orphan state, where they wait indefinitely to join other clusters while SAC devices form their own clusters.

C. Authenticating Cluster Advertisments

Like other cluster traffic, cluster advertisements are also protected by a MAC. Attackers cannot generate verifiable cluster advertisements unless they possess the valid cluster key. Furthermore they can not replay older advertisements because these contain a sequence number and duplicate advertisements are discarded by cluster devices. However compromised devices can generate illicit cluster advertisements with valid sequence numbers that can be verified by other cluster devices. The only possible aim of such an attack would be to hide the absence of the security agent from other cluster devices, possibly to stop them from abandoning the existing cluster. However this attack can not be successful indefinitely because the attacker can not create a verifiable re-key packet (Section VI.A). Additionally the negative affect of such an attack is limited since the attacker can launch much more deadly attacks such as polluting the routing information etc.

Since devices have no way of verifying advertisements from clusters to which they do not belong, attackers can easily generate fictitious cluster advertisements. However as devices always perform mutual authentication this is at best a denial of service attack. Devices can protect themselves against such false advertisements by limiting the rate at which they attempt to re-authenticate after recent failed attempts.

VI. KEY MANAGEMENT

During device authentication the security agent establishes a secure channel, verifies the supplicant device's credentials, and downloads the cluster key and policy to the supplicant. The cluster key is then periodically refreshed with the period whose duration depends on the security level required. In general, re-key messages can be sent unicast or broadcasted for more efficient distribution. We rejected the unicast approach because it is inefficient and does not utilize the properties of the broadcast medium.

Our approach also implements an efficient mechanism to evict short-term cluster members. Short-term cluster members are basically foreign devices that have been granted cluster membership for a limited duration. These could be devices belonging to friends and family or even rented devices that should be evicted from the cluster before returning. Cluster devices that belong to the user's own PN are considered long-term members. Evicting long-term members is a rare occurrence which for example needs to be done if a device gets sold or compromised. In such a case, the device's long-term security associations with the PN will be eliminated, followed by the security agent dismantling the existing cluster. When the cluster re-forms the evicted PN device will not have any long-term security associations to be able to authenticate itself.

A. Cluster Key Updates

Even though all cluster traffic is protected using a MAC, we believe that re-key messages containing the new cluster key require an additional level of security. This is necessary in order to reduce the impact of compromised devices which can otherwise hijack the cluster by pretending to be the security agent and updating the cluster key. Therefore broadcast re-key messages are protected using source authentication so that they can not be forged by compromised devices. Finally, to ensure confidentiality, the new key is distributed encrypted using the existing cluster key.

As we are broadcasting the re-key messages, there is no way for the security agent to ensure that all members receive the new cluster key. We can imagine a solution in which members can generate negative acknowledgements when the cluster key that they are using is about to expire and they have not received the new key. However devices can not send negative acknowledgements if the re-key is being done ahead of schedule, for example, to evict a shot-term member. Lastly, a reliable key update using pair-wise keys and acknowledged unicast transmission is possible but significant more costly and requires the security agent to have an up-to-date list of all cluster members.

Our re-key mechanism has two goals. The first goal is to ensure a reliable distribution of re-key messages to all cluster members. The second goal is to enable cluster members to verify the source of these messages. In order to do so we need a reliable broadcast authentication mechanism based on symmetric cryptography. The broadcast authentication mechanism we use has some similarities with μTesla [5], but ported to the peculiarities of our system. Notably we do not want to use clock synchronization, even the "loose" clock synchronization required by μTesla.

μTesla brings about asymmetry by a delayed disclosure of symmetric keys. It divides time into equal intervals, assigning a different key to each interval. All packets generated by the source within a specific time interval use the key assigned to that interval. Messages are then broadcasted with a MAC generated using the secret key, which will be disclosed at a specific time in the future. When the receiver receives a message, it confirms that the key has not been disclosed, and

then buffers the message. The message is later authenticated when the corresponding key is publicly disclosed.

Although our solution is also based on a delayed disclosure of symmetric keys (hash chain values), it is not tied to time intervals but rather to each round of the re-key algorithm. In our approach, re-key messages are protected with a MAC calculated using a symmetric key (hash chain value) related to that round of re-keying. In order to ensure the reliable delivery of a re-key message, it is encapsulated within "i" consecutive cluster advertisements; where "i" was defined as the number of sequential cluster advertisements that are missed by a device before it believes it has lost connectivity with the security agent. Note that the "complete" cluster advertisement carrying the encapsulated re-key message is protected with the MAC generated using K_{mac}. This means that even though re-key messages cannot be authenticated till the corresponding hash chain value is released, external attackers can not launch denial-of-service (DoS) attacks against the cluster. This is because they cannot generate valid cluster advertisements encapsulating the false re-key messages!

The hash chain value corresponding to this re-key message is then publicly released in the next "i" sequential cluster advertisements. As above, any device which has connectivity with the security agent should successfully receive this hash chain value. The value is then verified against the authenticated hash chain value given to all devices by the security agent when they joined the cluster. Thus re-key messages cannot be successfully attacked, even by compromised devices, because nobody but the security agent knows an undisclosed authentic hash chain value.

The cluster advertisement which includes the hash chain value being disclosed is not source authenticated, and is therefore open to attack by compromised devices. However, if such a device changes the original hash chain value (and updates the MAC of the cluster advertisement) then the disclosed value will not match the re-key message.

B. Evicting Short-Term Members

In order to evict short-term members the new cluster key can not be distributed by encrypting it in a KEK (Key Encrypting Key) that is known to both long and short-term members. Therefore unlike PN devices which are given the actual cluster key when they join the cluster, foreign devices are only given the derived cluster keys K_{mac} and K_{encr}. Although this allows them to create a valid MAC for authentication they are unable to decrypt the broadcasted re-key messages meant for long-term members that contain the updated cluster key. The security agent simultaneously updates the derived keys at the foreign device using unicast transmission. This does not create a performance drawback because the number of such short-term members is limited. The re-key messages meant for long-term members are for a much broader audience and are broadcasted for increased efficiency.

Consequently, when a short-term member needs to be removed from the cluster, the security agent updates the cluster key of all long-term members using the mechanism described in Section VI.A. It will then change the keys of the other short-term members (except the device that is being evicted) using secure unicast transmission.

C. Device Authentication

Earlier we stated that devices wishing to join a cluster need to authenticate with the security agent of that cluster. We also said that members of a cluster only forward authenticated cluster traffic. This implies that for supplicant devices to authenticate with a cluster, they need to be within the transmission range of the security agent (belonging to the cluster they wish to join). Similarly, for two clusters to merge together, the two security agents also need to be within each others transmission range. Such a restriction on the extensibility of the cluster is not practical. We would like clusters to extend with devices that are within the range of even peripheral cluster members. Similarly, two clusters should be able to merge when their periphery overlaps and not only when the transmission range of the two security agents overlaps. To that end, cluster members enable IEEE 802.1X based port authentication.

As a result, besides authenticated cluster traffic (i.e. traffic protected using K_{mac}) cluster members also accept unauthenticated EAP [11] requests which are forwarded to the security agent for authentication. As a result supplicant devices do not need to be within the communication range of the security agent to be able to authenticate themselves. Although allowing clustered devices to forward unauthenticated EAP requests make them vulnerable to DoS attacks, clustered devices can protect themselves by limiting the rate at which they forward such requests. Predictably, devices that are not part of a cluster do not forward EAP requests.

The mechanisms we propose for use have some important differences with IEEE 802.1X. For instance, after a successful authentication the supplicant no longer maintains any relationship with the authenticator. Additionally, our mechanisms allow complete clusters to merge instead of just permitting individual devices to join a cluster. When clusters merge, the authentication takes place between the two security agents using a secure tunnel and any confidential information exchanged is not visible to the intermediaries who are just forwarding the EAP messages. A detailed specification of the new EAP protocol is beyond the scope of this document.

For a cluster that is extended with a single device the process is simple. The new device configures itself according to the cluster policy and begins to use K_{mac} to take part in cluster communication. When two clusters merge, the security agent of one cluster needs to step down. This is done using a simple check to see which security agent has a higher "weight". Security agents calculate their weight, a numerical value that expresses their current status keeping in mind device parameters like mobility, battery level, number of devices in the cluster etc. When two clusters merge the yielding security agent needs to update all the devices in its cluster with the information it has received. As with cluster key updates, this must be done using source authentication. Therefore the mechanisms used are the same as that for re-keying.

D. Pre-Authentication

In order for two devices to authenticate each other they must have an existing security association, the creation of which precedes the actual authentication. As of now we have assumed that such an association already exists between the security agent and the device it is authenticating. Since security associations require some work to set up, we would like to reduce the total number of security associations required by our system. Our model assumes that all PN devices have a security association with the core node, a PN wide master node. This association is formed during the imprinting process, which each device goes through when it becomes a member of the PN. In order to reduce the number of security associations between devices and security agents, security agents can forward the authentication requests of unknown devices to the core node. The core node will authenticate the device and also assist in creating a long term security association between the security agent and the device. Therefore in the future the security agent will not have to proxy the authentication requests.

If connectivity with the core node is not available, or there is no security association between the authenticating device and the core node then the security agent can give the user an option of authenticating using manual intervention (like Bluetooth pairing).

VII. SIMULATIONS

The objective of our simulations was to quantify the cost of *initial* cluster formation and to optimize its performance by examining the effect of different parameters. In this first set of simulations, we study the formation of *one* cluster with *one* security agent. We do not (yet) look at key management, cluster merging or effects of mobility.

We simulate a CSMA/CD (802.11b [12]) wireless ad-hoc network with the wireless mobility extensions in ns-2.29 [13]. The transmission range of each device is 10m and simulations are carried out till all devices have authenticated with the security agent (to form a cluster). When not specified otherwise, the simulation uses default ns link layer parameters, for instance a RTS/CTS threshold of 0 bytes. Setting a higher RTS/CTS threshold value produced errors [14]; therefore the default value was used in all simulations. All our simulation scenarios use a grid size of either 25 x 25m or 50 x 50m, which correspond to average inter device distances of approximately 2 and 5 hops respectively. For each scenario (each row in Tables I and Table III) we generated 50 connected graphs with random device positions. The simulation of each of the 50 graphs was performed 10 times using a random seed. The results are summarized in the corresponding tables.

The scenarios differ in their grid size, the number of devices and/or the value of certain PN specific simulation parameters. The performance in each scenario is judged by the average time and the *total* transmissions at the MAC layer required for all the devices to complete authentication with the security agent. This gives a quantitative idea of the cost in terms of clustering delay and energy consumption. Ideally we would like to compare the security overhead to data traffic, but that can vary depending on the amount of data traffic.

TABLE I. SIMULATION RESULTS USING DEFAULT PARAMETERS

Grid Size (m)	No. of devices	Avg. Time (s)	Avg. Bytes (k)	Avg. SA Bytes (k)
25 x 25	5	1.3	35	12
25 x 25	10	2.8	108	26
25 x 25	20	4.0	244	54
25 x 25	30	4.4	353	83
25 x 25	40	5.0	461	111
25 x 25	50	5.3	561	140
50 x50	20	6.8	424	54
50 x 50	50	10.1	1378	138

TABLE II. PN SIMULATION PARAMENTERS AND THEIR DEFAULT VALUES

Simulation Parameter	Default Value
CA period	1s
CA packet size	44 bytes (incl. 4 bytes MAC, 20 bytes IP)
EAP packet size	534 bytes (incl. 4 bytes MAC, 20 bytes IP)
EAP Round Trips	4
EAP timeout	2.5s
Auth. timeout	20s
Auth. delay	0s
Processing delay of CA/EAP packets	Average of 10ms

The results in Table I are achieved using the default PN simulation parameters of Table II. When comparing the sixth and the eighth simulations of Table I, we see that the extra number of hops (avg. of 2 vs. 5) has a significant effect on the total transmission overhead. However, since the total number of devices authenticating with the security agent is the same in both scenarios the transmissions overhead of the security agent is identical.

Fig. 2 illustrates one simulation instance of a grid size of 25 x 25m with 20 devices. At time 10ms the security agent (SA) transmits the first cluster advertisement (CA). Every unauthenticated device that hears this CA will attempt to authenticate with the SA, i.e. all the devices within the dark grey circle around the SA. In the best case when there are no collisions at the link layer, all unauthenticated devices within the first hop will successfully authenticate with the SA. At 0.5s in Fig. 2 we see that 6 devices have completed authentication with the SA. Once authenticated, devices are able to validate the MAC on any future CAs they receive, and will therefore forward such CAs. In the best case we can expect devices which are two hops away from the SA to authenticate with the SA after the second CA is transmitted. As the default CA period is 1 second during the clustering phase, we see 5 more devices have authenticated at 1.5s in Fig. 2. The value of 1 second for the CA period is only meant to be used during the clustering phase to reduce the clustering delay and not through the lifetime of the cluster. At 2.5s, we see that two devices in spite of being within the transmission range of the third CA, are unauthenticated. This is due to collisions at the link layer; either they did not receive the CA, or the EAP packet they sent got lost. The chosen EAP authentication mechanism is simulated based on two parameters, the average size of the EAP packet and the number of round trips necessary to complete the authentication (default values in Table II). If an EAP message does not get a reply within the "EAP timeout"

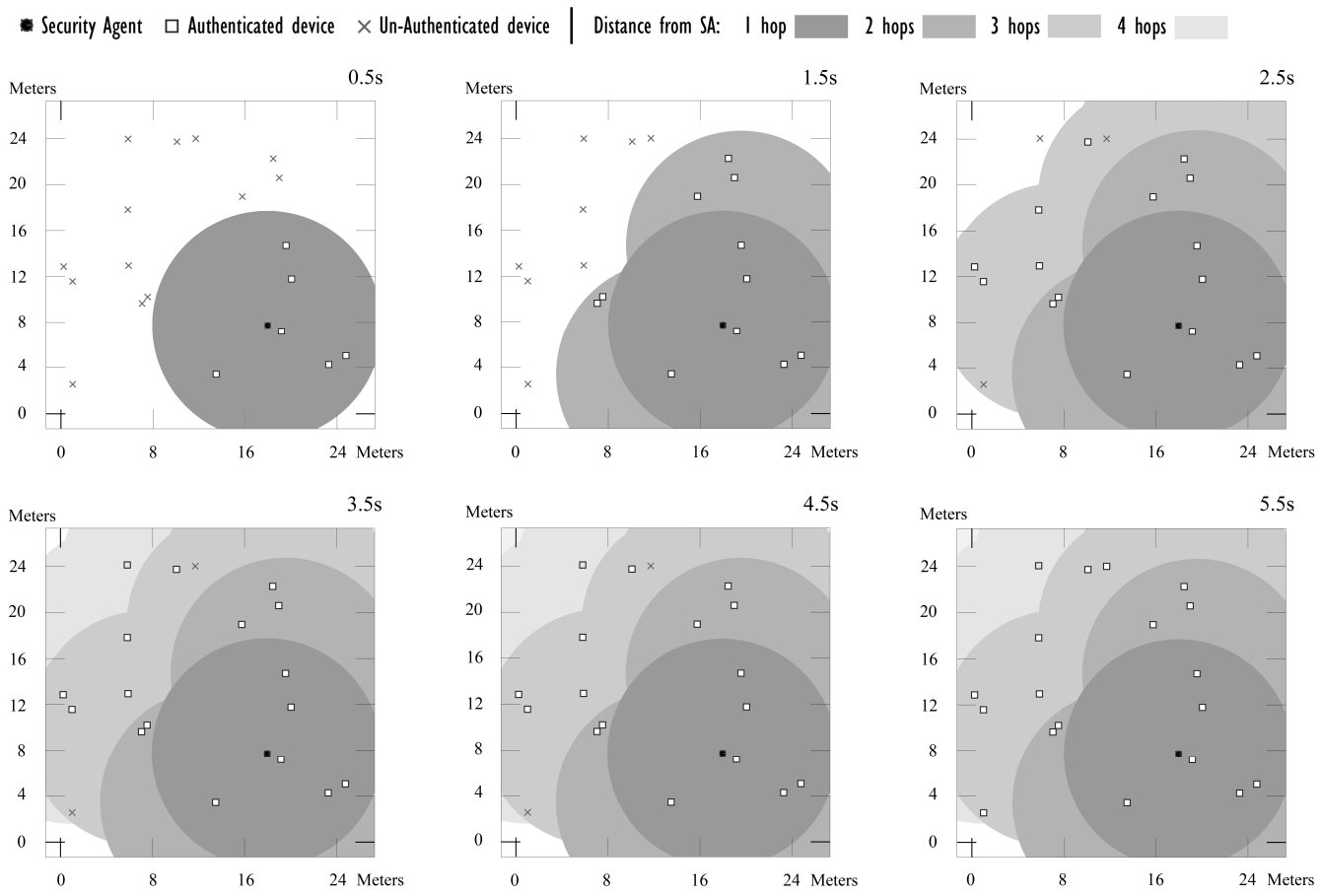

Figure 2. Simulation instance with a grid size of 25 x 25m and 20 devices

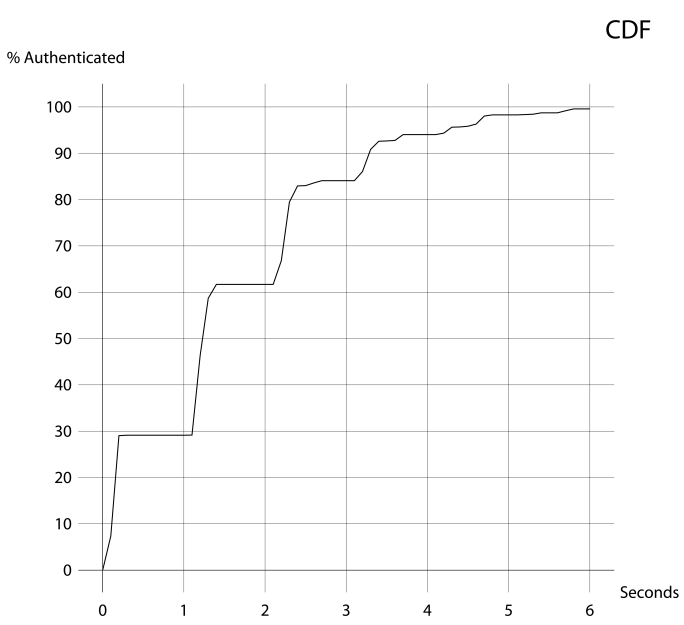

Figure 3. CDF for simulations with a 25 x 25m grid and 20 devices

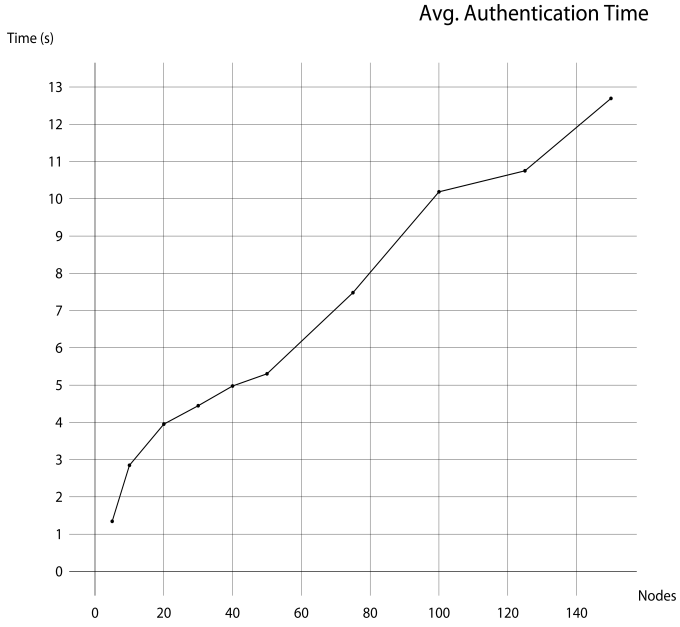

Figure 4. Number of devices vs. clustering delay for a 25 x 25m grid

period of 2.5s, it is retransmitted. That is why the two unauthenticated devices (which received their CA at ≈2s) only try again at ≈4.5s, and we see that they have successfully authenticated by 5.5s. Lastly, if the entire authentication does not complete within the "Auth timeout" period, devices give up and will only try again the next time they receive a CA

For each scenario of Table I, the time taken for devices to authenticate with the security agent (although strongly correlated with the CA period) varies across different simulation runs. This is due to the collisions at the link layer. Fig. 3 illustrates a CDF of time taken by devices to authenticate with the SA. The graph is compiled from data of all the simulations carried out for the given scenario i.e. 50 graphs x 10 runs each. We can see that although the average clustering delay for this scenario was 4.0s (Table I), over 60% of the devices have authenticated by 1.5s. As expected, new devices authenticate in each round of CAs (shown by large hikes at 0s, 1s, 2s, 3s etc). The small hikes at 2.5s, 3,5 and 4.5s correspond to authentications that require retransmission of EAP messages.

Fig. 4 illustrates how the time needed to complete cluster formation varies with the number of devices in the system. We can see that the increase is almost linear, likely because the benefit of having more devices authenticating per cluster advertisement is balanced by delay caused by the increased collisions at the link layer.

TABLE III. SIMULATION RESULTS FOR A 25 X 25M GRID WITH 50 DEVICES, USING MODIFIED SIMULATION PARAMETERS

EAP Timeout (s)	Auth. Delay (ms)	Avg. Time (s)	Avg. Bytes (k)	Avg. SA Bytes (k)
2.5	0	5.3	561	140
3.0	0	7.1	573	141
3.0	30	5.4	562	140
3.5	0	6.4	566	140

Table III illustrates the effect of tuning the "EAP timeout" and the "Auth delay" parameters. The first row corresponds to using the default values for the given parameters. In the second row, the EAP timeout is increased to 3.0s and as expected the clustering delay increases. This is because the system responds slower to lost EAP messages. However, when the timeout value is increased to 3.5s we see that the performance is better than for 3.0s. This is because with an "EAP timeout" of 3s any retransmission of EAP messages will happen while CAs are being disseminated, thus increasing the chance of collisions. We confirm this by observing a performance increase if there is a 30ms delay before the authentication phase begins.

As an optimization, when CAs are disseminated in the network, devices create temporary routing table entries to store reverse paths towards the SA. As a result, when device forward EAP messages to the SA, the complete path to the SA already exists and does not need to be discovered. The forward route to the sender is created when the first EAP message travels to the security agent. As a result, the MAC bytes calculated above do not have any routing over head. However, these optimizations are derived from the AODV implementation in *ns* and can only work when using AODV as the routing algorithm.

VIII. CONCLUSION

In this paper we have formulated a security architecture for Personal Networks. Since our mechanisms are solely based on fast symmetric cryptography, they are applicable to a wide variety of device types. Our design takes into account the peculiarities of the system, the most constrained of which is battery life. In this context we have specified the procedures for clustering, the functionality of the security agent, mechanisms used for cluster discovery and those for key management. Additionally we have carried out simulations to show the feasibility of our proposed security mechanisms.

ACKNOWLEDGMENT

This work was supported by the Dutch Ministry of Economic Affairs under the Innovation Oriented Research Program (IOP GenCom, QoS for Personal networks @ Home). We would like to thank all the members of the project for their discussions and contributions.

REFERENCES

[1] I. G. Niemegeers and S. M. Heemstra de Groot, "Research issues in ad hoc distributed personal networking," Wireless Personal Communications, vol. 26, no. 2-3, pp. 149–167, August 2003.

[2] http://qos4pn.irctr.tudelft.nl/

[3] Weidong Lu, Anthony Lo, and Ignas Niemegeers, "Research Issues in QoS Provisioning for Personal Networks", Thirteenth International Workshop on Quality of Service (IWQoS 200 5), Passau, Germany, 2005.

[4] X. Hong, M. Gerla, Y. Yi, K. Xu, and T. J. Kwon, "Scalable Ad Hoc Routing in Large, Dense Wireless Networks Using Clustering and Landmarks", IEEE International Conference on Communications (ICC 2002), New York City, NY, April 2002.

[5] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. "SPINS: Security protocols for sensor networks", The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001), Rome, Italy, July 16-21, 2001.

[6] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", The Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, Maryland, USA, November 3-5, 2004.

[7] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure Pebblenets", Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), October 2001.

[8] T. Hardjono, and L. R. Dondeti, Multicast and Group Security, Artech House Inc, Norwood, MA, 2003

[9] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication", 6th ACM Conference on Computer and Communications Security, November 1999.

[10] A. Perrig, R. Canetti, D. Song, and J.D. Tygar, "Efficient and Secure Source Authentication for Multicast", Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2001.

[11] B. Aboba, and D. Simon, "PPP EAP TLS Authentication Protocol", RFC2716, IETF

[12] http://www-ece.rice.edu/~jpr/ns-802_11b.html

[13] http://www.isi.edu/nsnam/ns

[14] http://www.dei.unipd.it/wdyn/?IDsezione=2435

[15] IST MAGNET, http://www.ist-magnet.org/.

[16] IST-507102 MAGNET/WP4.3/UNIS/D4.3.2/PU/1.0, "Final Architecture of the Network-Level Security Architecture Specification", March, 2005.