This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.

Evaluating Secure Cluster Formation in Personal Networks

A. Jehangir
University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
The Netherlands
jehangira@cs.utwente.nl

S. M. Heemstra de Groot
Twente Institute for Wireless and Mobile Communication, and
Delft University of Technology
The Netherlands
Sonia.Heemstra.de.Groot@ti-wmc.nl

***Abstract*—In this paper we evaluate our previously proposed security architecture for Personal Networks (PNs). Personal Network is a new concept utilizing pervasive and distributed computing to meet the needs of the user. We aim to secure Personal Networks with lightweight security mechanisms that are suitable for resource constrained devices yet robust enough for self organization and secure communication. In order to study the behavior of our proposed security mechanisms we developed a simulation environment in NS-2. The simulations are used to evaluate the overhead of our mechanisms and to understand the effects of key parameters. The results show that our mechanisms have low delay and energy requirements and are feasible for the heterogeneous devices we envision in our PN.**

***Index Terms*—Cluster formation, Personal Networks, Secure communication, NS-2 simulations**

I. Introduction

FUTURE mobile communication systems are envisioned to provide their users access to services anywhere and at anytime. **Personal Networks (PNs)** [1] have similar aspirations but take a very user-centric approach to solving the challenges. They comprise a core consisting of a **PAN** (Personal Area Network) which, when necessary, can be extended to include devices and services both in the vicinity and those at remote locations. This extension can be made using both infrastructure based networks and wireless networks to access the range of services available on and through these networks, including those offered by other PANs. PNs are composed of heterogeneous devices and are dynamic entities due to the mobile nature of their constituents and the spontaneous nature of their connectivity with the outside world. Figure 1 outlines the network layer abstraction of a PN in the QoS for PN@Home project [2]. We see personal devices organized in the form of **clusters**, which are groups of personal devices that can communicate amongst each other without using any non-personal devices. Clusters belonging to a user interconnect to form his PN.

This work was supported by the Dutch Ministry of Economic Affairs under the Innovation Oriented Research Program (IOP GenCom, QoS for Personal networks @ Home) and Freeband PNP2008 project. The authors would like to thank all members of the projects for their discussions and contributions.

As PNs edge closer towards reality, security becomes an important concern since any vulnerability in the system will limit its practical use. The goal of our security architecture is to provide users of PNs with a reliable communication platform to access their services. In [3] we gave an overview of a centralized architecture for securing personal clusters. We defined a new role for a device in the cluster, that of a **security agent**. In this paper we will evaluate the overhead of those mechanisms using simulations. Since most of the (energy) overhead arises from the transmission of extra data rather than computation costs [4], we are particularly interested in the transmission overhead of security.

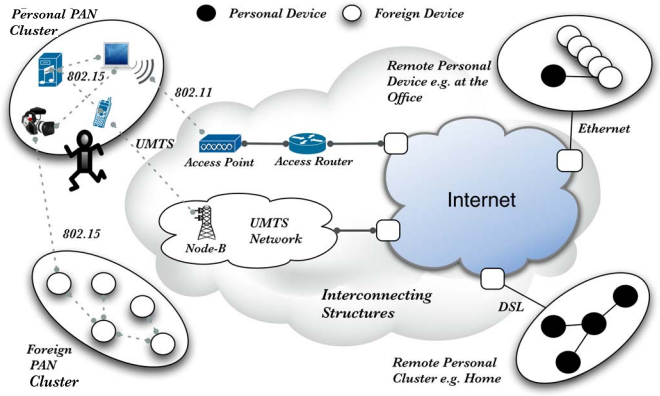

Figure 1: Network layer abstraction of a Personal Network

Due to the heterogeneous nature of devices within the Personal Network, our aim is to use secure but lightweight mechanisms suitable for resource constrained devices and wireless communication. In [3] we proposed pair-wise keys for secure cluster formation and group keys (called **cluster keys**) for securing intra-cluster communication. The cluster key is periodically refreshed by the security agent and distributed to all existing cluster members. It is also given to new members

by the security agent after a successful authentication. Clustered devices append a MAC (message authentication code) calculated using the cluster key to all cluster traffic that they generate. As a result, any subsequent clustered device can verify that the messages were generated by a trusted device and not modified in transit by any un-trusted devices.

The security agent periodically broadcasts **cluster advertisements** which are used by other devices to discover the cluster. In terms of security agent functionality we defined two new classifications of devices. **Security agent capable (SAC)** devices have the capabilities to function as security agents whereas **security agent in-capable (SAI)** devices do not. SAI devices e.g. sensors are less sophisticated and typically only useful in conjunction with other smarter devices, when networked together as a cluster.

The aim of this work is to evaluate secure cluster formation as originally proposed in [3]. In Section II of this paper we summarize some other work related to securing Personal Networks and in Section III we give an overview of the clustering process, focusing on issues related to its implementation in the simulator. Section IV discusses simulation aspects of cluster merging while Section V presents our simulation results. In Section VI we state our conclusions and sum up remaining future work.

II. Related Work

The security architecture proposed for Personal Networks in the MAGNET project [5] provides an interesting basis for comparison with our work. Unlike our centralized approach to cluster formation, MAGNET devices wishing to join a cluster require a security association with at least one neighboring clustered device. Since communication between neighboring devices is done using pair-wise keys, they then use the transitivity of trust to establish individual security associations with any neighbors with whom they do not have an existing security association. This results in a significant overhead, especially during device mobility. Furthermore using pair-wise keys to secure both unicast and broadcast communication between neighbors increases the processing overhead due to the hop-by-hop encryption and decryption necessary for messages traveling multiple hops. Our approach of using a group key avoids this problem and also reduces the overhead of key management significantly. Finally, the centralized nature of our security architecture can be an asset since it provides the user with a higher degree of control.

TinySec [6] is a lightweight link layer security architecture designed for sensor networks. It aims to provide reasonable security for devices with extreme resource constraints. It bears similarities to our proposed mechanisms in that both use group keys to provide message integrity, authentication and (optionally) confidentiality. The TinySec protocol has been well designed with respect to its energy usage, however its applicability in Personal Networks is limited since it is not a complete security framework e.g. it does not formulate any mechanisms for key management.

III. Clustering

Devices that are within each others transmission range, and belong to the same PN organize themselves in the form of clusters. The first cluster is always created when a SAC device starts functioning as a security agent. This cluster (consisting of just the one device) is then extended by incorporating other devices or even other clusters.

Our PN simulation scenario is composed of SAI and SAC devices which are derived from, and extend the functionality of NS-2 [7] mobile nodes. SAI devices initialize into the *orphan* state while SAC devices start out functioning as security agents in a cluster that only contains them. Since SAC devices start out as security agent they are fully initialized with a cluster key, **PN id**, **cluster id** etc. The PN and cluster ids [3] are needed to distinguish between traffic belonging to different PNs and/or clusters. The PN id is configured for all PN devices during device imprinting. During this imprinting phase the **core node**, a personal device that plays the role of the "mother" [8] creates the necessary long term security association with all new devices. Devices in the *orphan* state do not belong to any cluster and thus do not have a cluster key or cluster id etc.

Devices in the orphan state attempt to authenticate with the first cluster of their PN whose advertisement (i.e. with a matching PN id) they receive. After a successful authentication they join that cluster and their state changes to *clustered*. At this point, they have received the new cluster key (and its validity period), cluster id and cluster policy from the security agent. The cluster policy contains information that devices need to know in order to act appropriately.

A. Cluster Advertisements

Security agents advertise themselves periodically by broadcasting cluster advertisements which are used by other devices to discover the cluster. Other cluster members rebroadcast non-duplicate advertisements, in effect propagating them to the edges of the cluster. Figure 2 illustrates the packet format for a cluster advertisement without any extension headers. The *sequence number* field is incremented for each new cluster advertisement; it is used by receiving devices to determine, and drop duplicates. As in IPv6 each extension header (Figures 3-5) adds optional information to a cluster advertisement and is identified by a distinct *next header* value. A cluster advertisement may carry zero, one, or more extension headers, each identified by the *next header* field of the preceding header. Cluster advertisements also include the address of the sending security agent; it is used by devices to authenticate with the cluster. The purposes of the remaining fields of Figure 2 are described in Section C.

The re-key extension header (Figure 3) is used to update the existing cluster key when it is about to expire. During cluster

merging the merge extension header (Figure 4) is used *together* with the re-key extension header. This is because re-keying is a subset of cluster merging. After the new key has been transmitted, the security agent uses the hash disclosure extension header (Figure 5) to disclose the hash chain value used to calculate MAC-H on the re-key and merge extension headers. The fields in Figures 3-5 are described in Section D. Note that like every Layer 2 frame transmitted by clustered devices, the Layer 2 frame containing the cluster advertisement (with the extension headers) is also protected with a MAC generated using the cluster key.

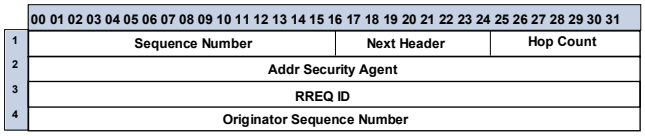

Figure 2: Cluster Advertisement header.

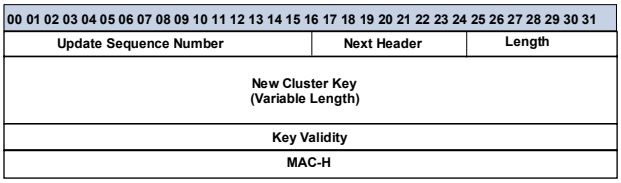

Figure 3: Re-Key extension header.

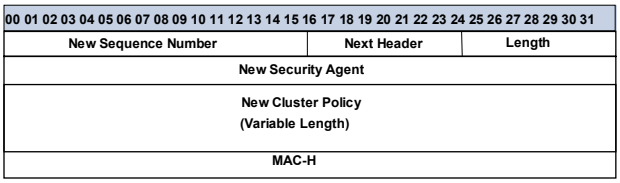

Figure 4: Merge extension header.

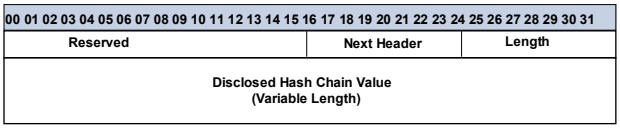

Figure 5: Hash disclosure extension header.

B. *Authentication*

Un-clustered devices belonging to the same PN as the cluster whose advertisement (without any extension headers) they receive, cache it and attempt to authenticate with the advertising security agent. If the authentication fails the cached cluster advertisement is discarded. On the other hand if the authentication is successful, the cached cluster advertisement is re-broadcasted.

Un-clustered devices ignore cluster advertisements containing extension headers because such cluster advertisements are an indication that the cluster key is being updated. Device will wait for the re-key phase to complete before authenticating with the security agent otherwise they may receive the old cluster key. For this reason security agents entering the re-key phase cancel ongoing authentications and reject new authentication attempts till the re-key phase completes.

Finally, caching the cluster advertisement during authentication and retransmitting it afterwards reduces the total clustering delay since all cluster-able devices can be incorporated into the cluster after only one CA, instead of incorporating the first hop device after the first cluster advertisement and the second hop devices after the second CA, etc. As a result the cluster formation overhead is much lower than presented previously [3].

Since we would like clusters to extend with devices that are outside the range of the security agent but within the range of peripheral cluster members, cluster members enable IEEE 802.1X based port authentication. Therefore, clustered devices accept authentication requests from other devices (i.e. without a valid MAC) which are forwarded/tunneled to their security agent for verification. These authentication requests are typically in response to cluster advertisements previously transmitted by the clustered devices. Since these authentication requests cannot be authenticated before being forwarded, clustered devices will need to limit the amount of authentication requests that they forward in any given period in order to guard against DoS attacks.

During authentication the security agent transfers required cluster parameters to successful authenticators. We can not specify the exact format of an authentication packet because that is dependent on the EAP mechanism [10] used. Finally, all authentications must complete within *auth-timeout* (Table 1) and each EAP message (transmitted over UDP) must get a reply within *rtn-timeout* (Table 1) or it is retransmitted.

C. *Reducing Path Discovery Overhead*

When a security agent (SA) transmits a cluster advertisement its AODV module populates the *hop count*, *RREQ id* and *originator sequence number* fields of Figure 2. These fields have the same functionality as their namesakes in an AODV RREQ packet and are used by forwarding devices to create temporary routing table entries to store reverse paths towards the SA.

When a device receives a cluster advertisement from its security agent that has not been processed before, it updates the *hop count* value and searches for a reverse route to the security agent in its routing table. Depending on the result, the route is either created or updated using standard AODV behavior e.g. the lifetime of the reverse route entry is set using default parameters. The cluster advertisement is then processed by the device and re-broadcasted. Consequently, when the ensuing authentication requests need to be forwarded to the SA, the complete path to the security agent already exists and does not need to be discovered. The forward path to the authenticator is created when its first authentication packet travels to the SA.

The authenticator adds the following three fields to the first authentication packet that it sends to the security agent; *hop count*, *destination sequence number* and path *lifetime*. As

intermediate devices forward the returning authentication packets to the security agent, they use the information existing in the packet to create/update the forward route to the authenticator. The processing of the fields is the same as that of an AODV RREP packet. We can think of cluster advertisements like AODV RREQ packets that are disseminated in the entire network, and the resulting authentication packets as the AODV RREP packets that are unicasted back to the SA.

For this optimization, we only use a subset of AODV features that allow intermediate devices to create the freshest/shortest route to the SA. Since the cluster advertisement is not meant for any specific device it does not use all the fields of a typical AODV RREQ packet, such as the *destination address* and *destination sequence number* etc. Also, further features of the AODV implementation such as the expanding ring and exponential back off etc. are not necessary since we need network-wise dissemination of cluster advertisements and because the security agent does not wait for any reply to its advertisement. Lastly, since these optimizations are derived from the AODV implementation in NS-2 they require using AODV as the ad-hoc routing algorithm for the simulations.

D. Updating the Cluster Key

Updating the cluster key (or re-keying) is necessary when the current cluster key is about to expire, or when one cluster has to update its key to that of another cluster during cluster **merging**. Cluster merging is the process whereby two clusters of the same PN (within each others transmission range) merge to form one cluster. 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. Figure 6 illustrates our proposed re-key mechanism where re-keying starts with the cluster advertisement (including the re-key extension header) of sequence number **`x`**.

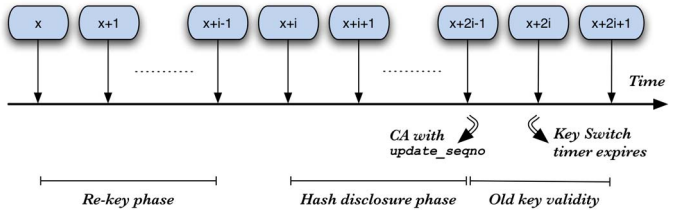

Figure 6: Re-key mechanism.

In order to ensure the reliable delivery of re-key information it is sent within **`i`** consecutive cluster advertisements, where the redundancy factor **`i`** is 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. A larger redundancy factor increases the reliability of message dissemination, but conversely costs more in terms of transmission overhead and delay. In Section V we will use our simulation results to justify a value for **`i`**.

The re-key extension header (Figure 3) sent in the re-key phase holds the *new cluster key* encrypted using the existing cluster key, the *key validity* period of the new key and the *update sequence number*. The *update sequence number* field of the re-key message holds the value of the sequence number of the cluster advertisement (illustrated in Figure 6) which signals the switch over to the new key. Additionally, the fields of Figure 4 contain information that merging devices need to know in order to become part of the new cluster.

The re-key extension header is protected using MAC-H generated using a secret key (hash chain value) which will be disclosed by the security agent during the hash disclosure phase. Therefore device buffer re-key messages till hash chain value corresponding to that round of re-keying is released. Using the property of a hash chain, the *disclosed hash chain value* (Figure 5) is then authenticated by verifying it against the hash chain value used in the last round of re-keying (also given to new cluster members by the security agent when they join the cluster).

For a device to re-key successfully it should receive at least one copy of the re-key message and one copy of the hash disclosure message. Thus, as per our earlier definition of **`i`**, any device that is a member of the cluster should be able to re-key successfully. If an SAI device fails to re-key, it will eventually go back to the orphan state, while an SAC device will re-initialize to form a new cluster.

IV. Cluster Merging

PN devices forward all authenticated traffic from fellow cluster members, and drop all un-authenticated traffic from external devices except authentication requests and cluster advertisements. In Section III.B we explained that authentication requests are forwarded by cluster members to their security agents for verification. Similarly, cluster advertisements belonging to another cluster of the same PN i.e. with a similar PN id but different cluster ids, are also forwarded to the security agent of the cluster. The security agent will then attempt to authenticate with its advertising counter part, in order to merge the two clusters. Such a mechanism is necessary for clusters to be able to merge when their periphery overlaps and not only when the transmission range of the two security agents overlaps. Since clusters belonging to different PNs can not merge, clustered devices drop advertisements of other PNs.

A. Forwarding Un-authenticated Cluster Advertisements

As cluster advertisements from one cluster can not be authenticated by forwarding devices of another cluster, it is conceivable that an attacker would generate false cluster advertisements with the same cluster id in order to launch a DoS attack. Devices can limit the effect of such an attack by controlling the rate at which they forward unauthenticated

advertisements. Furthermore, intermediate devices do not forward multiple copies of the same cluster advertisement (received from multiple sources) to their security agent.

B. Authentication between Security Agents

Security agents are able to authenticate with multiple SAI devices at the same time, but only with one other security agent at a time. This is because authenticating with another security agent may result in cluster merging. After a successful authentication the security agent with the lower weight (a dynamic value representing the feasibility of a device to be a security agent) will yield to the other. However the security agent with the lower *weight* will only yield if the cluster key of the other cluster is valid for more than 6 * ***i*** * *cluster advertisement period*. This is because one re-key process always takes less than 3 * ***i*** * *cluster advertisement period* (Figure 6) and the new cluster key should be valid long enough for the merging security agent to update its own cluster key and for the newly merged devices to be able to receive any re-key packets of their new cluster. Predictably, security agents in the merging state reject any authentication attempts.

V. SIMULATIONS

The objective of our simulations is to (a) understand the effects of certain parameters on system performance and (b) get quantifiable values for the overhead of our proposals. For (a) we carry out simulations to study the effect of the redundancy factor ***i*** on the behavior of our system. For (b) will look at the cost of the following mechanisms: cluster formation with one or more SAC devices, control traffic for cluster maintenance, cluster key updates and cluster merging.

The NS-2 simulation script is used to create PN devices and initialize their parameters such as the device type, initial position, movement scenario and the PN id. Only devices with the same PN id can organize themselves to form a cluster. SAC devices must also be assigned a static *weight*, which is used during cluster merging.

We simulate a CSMA/CD (802.11b) wireless ad-hoc network with the wireless mobility extensions of NS-2.29. The network size is 50m x 50m and the transmission range of each device is 10m. For each simulation we use graphs with 5, 20 and 50 devices in order to cover a wider range of possible scenarios. For each of the three scenarios we generate 30 *connected* graphs with *random* device positions. Additionally, the simulation for each of the 30 graphs is performed 3 times using a random seed, for a total of 90 simulation runs per scenario. The overhead of our proposed mechanisms in each simulation is judged by the average time and total transmissions at Layer 2 required to organize the cluster. This gives a quantitative idea of the cost, in terms of delay and energy consumption.

A. Background Traffic

Our background traffic is audio and video streaming, something which we believe is typical for a PNs. Since the amount of background traffic has a significant effect on the optimal value of the redundancy factor, we simulate three types of background traffic: low, medium and high. Low background traffic corresponds to 128kbps, 44.1MHz MP3 audio streaming. We assume that each payload contains 3 MP3 packets of 417 bytes each, for a total of 1251 bytes. We use a CBR (Constant Bit Rate) traffic generator with a packet interval of 80ms. Medium traffic corresponds to low traffic plus an MPEG2 video stream of 256 kbps. The payload size is 1024 bytes and the CBR traffic generator has a packet interval of 32ms. High traffic corresponds to low traffic plus an MPEG2 video stream of 512 kbps. Besides packet loss due to collisions we can also have packet loss due to interference. Therefore devices in the system are assigned a packet error rate of 2% for all incoming traffic.

All background traffic is generated by foreign devices. Foreign devices do not belong to the PN so their traffic only competes with clustered traffic for contention of the transmission medium. Using clustered devices to generate background traffic is not ideal because they can only communicate after the cluster has formed, so the effect of background traffic on cluster (re)formation is limited. Foreign devices are placed equally spaced in a grid, the sender for each stream of the background traffic is chosen randomly from the bottom row and receiver from the top, thus ensuring that the background traffic traverses the network.

B. Choosing the right redundancy factor

Table 1 shows the values of the parameters used in the simulations. Authentication between devices is simulated by successfully exchanging four authentication packets in each direction, within the *auth-timeout* period. The parameter which has the largest effect on performance is the value of the redundancy factor ***i***. As explained earlier we use redundant transmissions to ensure a reliable distribution of broadcasted cluster advertisements and re-key messages.

TABLE 1: DEFAULT VALUES OF SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
cluster advert pkt size	40 bytes	cluster advert period	5s
Re-key ext. hdr	28 bytes	***i***	4
Merge ext. hdr	28 bytes	Key timeout	600s
Hash ext. hdr	20 bytes	auth-timeout	15s
Auth pkt size	534 bytes	rtn-timeout	3s

For simulations with one re-keying phase, Tables 2 through 4 show the average number of devices that are unable to re-key successfully due to lost re-key messages. Next to each value in the table (in brackets) is its standard deviation.

TABLE 2: LOST RE-KEY MESSAGES WITH LIGHT BACKGROUND TRAFFIC

# of Devices	$i = 1$	$i = 2$	$i = 3$	$i = 4$
5	0.19 (0.71)	0 (0)	0 (0)	0 (0)
20	1.09 (3.19)	0.1 (0.94)	0 (0)	0 (0)
50	0.93 (2.61)	0 (0)	0 (0)	0 (0)

TABLE 3: LOST RE-KEY MESSAGES WITH MEDIUM BACKGROUND TRAFFIC

# of Devices	$i = 1$	$i = 2$	$i = 3$	$i = 4$
5	0.57 (1.16)	0.04 (0.29)	0 (0)	0 (0)
20	3.72 (5.64)	0.34 (1.54)	0.27 (1.55)	0 (0)
50	4.54 (10.4)	0.99 (4.61)	0.06 (0.52)	0 (0)

TABLE 4: LOST RE-KEY MESSAGES WITH HEAVY BACKGROUND TRAFFIC

# of Devices	$i = 1$	$i = 2$	$i = 3$	$i = 4$
5	0.88 (1.48)	0.14 (0.55)	0 (0)	0 (0)
20	6.73 (6.85)	3.19 (5.11)	0.89 (2.66)	0 (0)
50	8.24 (13.5)	3.00 (9.06)	1.11 (4.11)	0 (0)

As expected, with low background traffic there is less contention for the transmission medium and a lower redundancy factor is sufficient to ensure reliable delivery. However, there are opposing effects as the devices in the system increase. On the one hand there is more contention in the system but on the other a higher device density means more redundancy. It is interesting too see that with light background traffic a cluster with 20 devices actually has slightly more re-key failures than that with 50 devices. This can be explained by the fact that with 20 devices spread over the same area as 50, there are fewer duplicate paths between a device and its security agent. So if one device fails to re-key it affects other devices that can only be reached through it. The relatively high standard deviation in the results of Tables 2-4 is because although most simulations have zero or one re-key failures, others in which a group of devices has connectivity through one device, have significantly more. To conclude, since we would like re-key failures to be probabilistically insignificant while keeping the overhead as low as possible, we choose $i = 4$ as the value of the redundancy factor for all further simulations. Furthermore, since the amount of background traffic has a smaller effect on our remaining simulations, we only use medium traffic for the remainder of the simulations.

C. *Cluster formation overhead*

We simulate cluster formation with one and with potentially five SAC devices. Table 5 shows the average cluster formation overhead with 95% confidence interval for a cluster with 5 devices. The overhead field measures the average sum of Layer 2 transmission by all PN devices to create the cluster.

TABLE 5: CLUSTER FORMATION OVERHEAD FOR 5 DEVICES

	1 SAC, 4 SAI devices	5 SAC devices
Duration (s)	2.1 ± 0.5	51.4 ± 3.9
Overhead (KB)	42.9 ± 2.3	50.2 ± 2.7

When there is only one SAC device in the system, the cluster is formed rather quickly. After the security agent advertises itself, all devices within the first hop range authenticate (in parallel). As soon as they have authenticated, they retransmit the cached cluster advertisement allowing devices which are two hops away from the security agent to authenticate, and so on. With 5 SAC devices, each device starts out being a security agent. Therefore each authentication can potentially result in cluster merging depending on if the security agent that is stepping down after merging has any clustered devices. If the security agent that is stepping down has not authenticated any devices to be part of its cluster, then it skips the cluster re-key process and the merge is instantaneous. The slight increase in the overhead between the two scenarios is the result of rejected authentication attempts, which need to be repeated. Some authentication attempts are initially rejected because a security agent can only authenticate with one security agent at a time. Table 6 shows the average cluster formation overhead with 95% confidence interval for a cluster with 20 devices.

TABLE 6: CLUSTER FORMATION OVERHEAD FOR 20 DEVICES

	1 SAC, 19 SAI devices	5 SAC, 15 SAI devices
Duration (s)	6.5 ± 0.8	81.8 ± 5.0
Overhead (KB)	450.1 ± 21.7	367.4 ± 20.2

As expected, with 1 SAC device the time taken for 19 devices to authenticate is more than that for 4 devices (Table 5). This is because a larger number of devices increase the probability of having a device more hops away. Generally devices within the first hop will authenticate first (then retransmit the cached cluster advertisement), next the devices within the second hop and so on. Similarly with the increase in the number of authentications and the more hops they need to traverse to get to the security agent, the total number of bytes transmitted increases non-linearly.

From Table 6 we can see that although the time taken for cluster formation with multiple SAC devices is larger due delay resulting from cluster merging, the transmission overhead is actually lower. This is because although the total number of authentications in the system remains the same, the number of hops between the authenticating device and the security agent it is authenticating with is actually reduced. Devices authenticate with the first security agents whose advertisement they receive, often this is the closest one in terms of the number of hops. As clusters grow, they overlap and merge after the two security agents authenticate each other. In this way instead of multiple devices authenticating with the security agents over several hops, there is only one *aggregated* authentication. This effect is more pronounced here when compared with simulations of 5 devices because of the increased number of hops in the system. Table 7 shows the average cluster formation overhead with 95% confidence interval for a cluster with 50 devices.

TABLE 7: CLUSTER FORMATION OVERHEAD FOR 50 DEVICES

	1 SAC, 49 SAI devices	5 SAC, 45 SAI devices
Duration (s)	11.1 ± 1.0	82.1 ± 4.0
Overhead (KB)	1440 ± 83	1099 ± 64

With simulations of 20 and 50 devices, the time needed to complete cluster formation with multiple SAC devices is larger than the simulation of 5 devices. This is because in the later simulation as there are no SAI devices, each cluster merge does not require a cluster key update by the security agent that is stepping down. Furthermore, with 50 devices in the system the average number of hops between two random devices is greater, therefore the benefit of aggregated authentication due to cluster merging is more pronounced (24% in Table 7 compared to 18% in Table 6).

D. Cluster maintenance overhead

Due to the centralized nature of our proposed security mechanisms the system needs to synchronize periodically. We proposed using periodic cluster advertisements to announce the presence of the security agent. Our simulations confirm that the security agent transmits one cluster advertisement (without any extension headers) of 103 bytes at Layer 2, per cluster advertisement period of 5 seconds. This cluster advertisement is then re-broadcasted once by each cluster member.

E. Re-keying overhead

During the re-key phase security agents transmit 4 cluster advertisements with the re-key extension header (131 bytes) followed by 4 cluster advertisements disclosing the hash chain value (123 bytes) used to protect the re-key messages. This comes out to be 1016 bytes per security agent and each clustered device; since they retransmit every cluster advertisement they receive. With the cluster advertisement period of 5 seconds, re-keying takes exactly 35 seconds to complete. Table 8 summarizes the transmission overhead of updating the cluster key with 95% confidence interval.

TABLE 8: RE-KEY OVERHEAD

	5 devices	20 devices	50 devices
Overhead (KB)	4.9 ± 0.1	18.5 ± 0.3	49.7 ± 0.6

As expected the growth is linear, since each device merely re-broadcasts the cluster advertisements it receives. The results are not exact factors of 1016 due to lost cluster advertisements resulting from interference and collisions.

F. Cluster merging overhead

This simulation is carried out by creating two identical but unconnected graphs, one of which then moves within the transmission range of the other. Three sets of results were obtained, for cluster merging between two clusters with 5, 20 and 50 devices each. The results are shown below.

TABLE 9: CLUSTER MERGING OVERHEAD

	5 devices	20 devices	50 devices
Duration (s)	35.9 ± 0.3	36.4 ± 0.3	37.1 ± 0.5
Overhead (KB)	28.9 ± 2.0	99.0 ± 4.1	220.6 ± 42.5

Once the two clusters come within transmission range the time taken for them to complete merging is dependant on the re-key period of 35s (Section E.). However, as the number of devices in the cluster increase, the number of hops between the two security agents (as well as the probability of packet loss over this longer path) increases. That is why as the size of the cluster increases we see an increase in the time needed for cluster merging. Furthermore, the overhead of cluster merging increases linearly as the devices in the cluster increase, again correspond to the results of Section E.

VI. CONCLUSION AND FUTURE WORK

In this paper we have validated our previously proposed proposals for securing Personal Network clusters. Since our mechanisms are based on fast symmetric cryptography they are applicable to the heterogeneous devices we envision in our PN. Moreover since most of the (energy) overhead arises from the transmission of extra data rather than computation costs we show that our design minimizes the transmission overhead of adding security by reducing the number of messages that need to be exchanged.

For future work we plan to investigate lightweight mechanisms for securing communication between different clusters of a Personal Network. Additionally, we would like to explore methods to access non-personal services in a secure manner and to create groups of Personal Networks that can cooperate.

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] QoS for Personal Networks at Home, http://qos4pn.irctr.tudelft.nl/
[3] A. Jehangir, S. M. Heemstra de Groot, "A Security Architecture for Personal Networks", *First International Workshop on Personalized Networks (PerNets 2006)*, San Jose, California, USA, July, 2006.
[4] 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 2001.
[5] MAGNET, http://www.telecom.ntua.gr/magnet/objectives.html
[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 2004.
[7] Network Simulator: NS-2, http://www.isi.edu/nsnam/ns
[8] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks", *7th International Workshop on Security Protocols*, April 1999.
[9] A. Jehangir, S. M. Heemstra de Groot, "Securing Personal Network Clusters", In submission.
[10] Extensible Authentication Protocol, http://www.ietf.org/rfc/rfc3748.txt