

University of Dayton eCommons

Computer Science Faculty Publications

Department of Computer Science

1-2009

A Secure Group Communication Architecture for Autonomous Unmanned Aerial Vehicle

Adrian N. Phillips

Air Force Institute of Technology

Barry Mullins

Air Force Institute of Technology


Richard Raines

Air Force Institute of Technology

Rusty O. Baldwin

University of Dayton, rbaldwin1@udayton.edu

Follow this and additional works at: https://ecommons.udayton.edu/cps_fac_pub

 Part of the [Graphics and Human Computer Interfaces Commons](#), and the [Other Computer Sciences Commons](#)

eCommons Citation

Phillips, Adrian N.; Mullins, Barry; Raines, Richard; and Baldwin, Rusty O., "A Secure Group Communication Architecture for Autonomous Unmanned Aerial Vehicle" (2009). *Computer Science Faculty Publications*. 118.

https://ecommons.udayton.edu/cps_fac_pub/118

This Article is brought to you for free and open access by the Department of Computer Science at eCommons. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of eCommons. For more information, please contact frice1@udayton.edu, mschlangen1@udayton.edu.

A secure group communication architecture for autonomous unmanned aerial vehicles^{††}

Adrian N. Phillips, Barry E. Mullins*[†], Richard A. Raines and Rusty O. Baldwin

Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433, USA.

Summary

This paper investigates the application of a secure group communication architecture to a swarm of autonomous unmanned aerial vehicles (UAVs). A multicast secure group communication architecture for the low earth orbit (LEO) satellite environment is evaluated to determine if it can be effectively adapted to a swarm of UAVs and provide secure, scalable, and efficient communications. The performance of the proposed security architecture is evaluated with two other commonly used architectures using a discrete event computer simulation developed using MATLAB. Performance is evaluated in terms of the scalability and efficiency of the group key distribution and management scheme when the swarm size, swarm mobility, multicast group join and departure rates are varied. The metrics include the total keys distributed over the simulation period, the average number of times an individual UAV must rekey, the average bandwidth used to rekey the swarm, and the average percentage of battery consumed by a UAV to rekey over the simulation period. The proposed security architecture can successfully be applied to a swarm of autonomous UAVs using current technology. The proposed architecture is more efficient and scalable than the other tested and commonly used architectures. Over all the tested configurations, the proposed architecture distributes 55.2–94.8% fewer keys, rekeys 59.0–94.9% less often per UAV, uses 55.2–87.9% less bandwidth to rekey, and reduces the battery consumption by 16.9–85.4%. Published in 2008 by John Wiley & Sons, Ltd.

KEY WORDS: group key management; multicast; security; unmanned aerial vehicles

1. Introduction

A swarm of autonomous unmanned aerial vehicles (UAVs) has great potential to provide benefits in a variety of applications, especially in the Department of Defense (DoD) intelligence, surveillance, and reconnaissance (ISR) mission. UAV swarm applications include continuous border patrol, battlespace surveillance, mapping routes for troop movement, real-time

information distribution to mobile military units, and extending communications via an airborne network. Grouping UAVs into a swarm allows them to carry a range of sensors with an array of capabilities, creating a diverse group that provides a wide viewing range and increased reliability through redundancy [1].

A swarm of UAVs is an example of a mobile ad hoc network (MANET). A MANET is a system of mobile hosts connected by wireless links, the union of which

*Correspondence to: Barry E. Mullins, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433, USA.

[†]E-mail: Barry.Mullins@afit.edu

^{††}This article is a U.S. Government work and is in the public domain in the USA.

forms a communication network modeled in the form of an arbitrary communication graph [2]. The lack of fixed infrastructure and the mobility of nodes in a MANET cause frequent changes in the network topology. This unpredictable environment presents numerous challenges especially in providing efficient communication. In addition, most nodes in a MANET rely on batteries, which limit available power and further compounds the challenge and increases the need for an efficient communication method.

One effective way to achieve efficient communication in a MANET is through multicast communication. Multicasting is a set of technologies that allows a source node to send data to multiple destination nodes simultaneously while transmitting only a single copy of the data on to the network [3]. The data are replicated for the destination nodes only when necessary. In a cooperative environment, such as a UAV swarm, nodes often need to transmit data to the entire group. Communication via multicasting, as opposed to unicasting, significantly reduces the processing load on the source and the overall bandwidth used in the network. Moreover, in a wireless environment, due to the broadcast nature of the wireless medium, multicasting has the potential to further reduce network traffic, and hence reduce network energy expenditure [4].

In addition to making communication efficient, ensuring the security of the communication is another increasingly important feature in a UAV swarm; this is especially true in the military. Previous UAV swarm research improved communication efficiency and effectiveness, with little emphasis on security [5–7]. However, the sensitivity of UAV swarm applications necessitates a secure communication architecture that provides DoD-mandated information assurance. With this added security component, a swarm of autonomous UAVs can provide a unique and powerful net-centric asset to support the warfighter.

2. Security in a Multicast Environment

2.1. Security Services

There are basic security services that should be built in all security architectures regardless if the environment is wired, wireless, unicast, multicast, infrastructure, or ad hoc. These include confidentiality, availability, integrity, non-repudiation, and authentication. In addition to these, there are security services that are unique to a multicast environment including group key secrecy, forward secrecy, and group access control (GAC). Group key secrecy guarantees that it is computationally

infeasible for an adversary to discover any group key [8]. Forward secrecy ensures new members are not able to read past traffic, and backward secrecy ensures former members are not able to read present and future traffic [9]. GAC is the ability to permit or deny membership into multicast groups [10].

2.2. Group Key Management

For secure wireless multicasting, cryptography and key management schemes are needed, in which cryptographic keys are used to encrypt and decrypt messages. The group key management scheme accomplishes the management and distribution of these keys. This includes enforcing the security services described above, which ensure only legitimate members of the multicast group hold valid keys and can access group data at any time during a multicast session.

In a UAV swarm, most if not all of the UAVs will be powered by batteries, making computationally intense exponentiations, such as public key cryptography, infeasible. An alternative solution is to implement symmetric key cryptography to secure communications. Symmetric key cryptography requires all group members to use the same shared key. This shared decryption key is often called the Session Encryption Key (SEK) or Traffic Encryption Key (TEK). To preserve the secrecy (both forward and backward) of the multicast data, the SEK needs to be updated upon certain events such as a member joining and leaving the group.

Group key management is one of the most resource-intensive operations on the network, and the dynamic nature of MANETs increases the complexity and overhead of managing this process. This is even more of a challenge when the number of members in the multicast group is large. Increasing the number of members not only increases the amount of keys that need to be distributed, but it also increases the frequency of rekeying because there will undoubtedly be more activity. This security overhead can overwhelm the network if a proper security architecture is not in place. Thus, the scalability of the selected security architecture is crucial when the size of the group grows.

3. Secure and Scalable Group Communication Architectures

The Hubenko architecture is a secure group communication architecture that combines the key features of the well-known multicast architectures in a way that

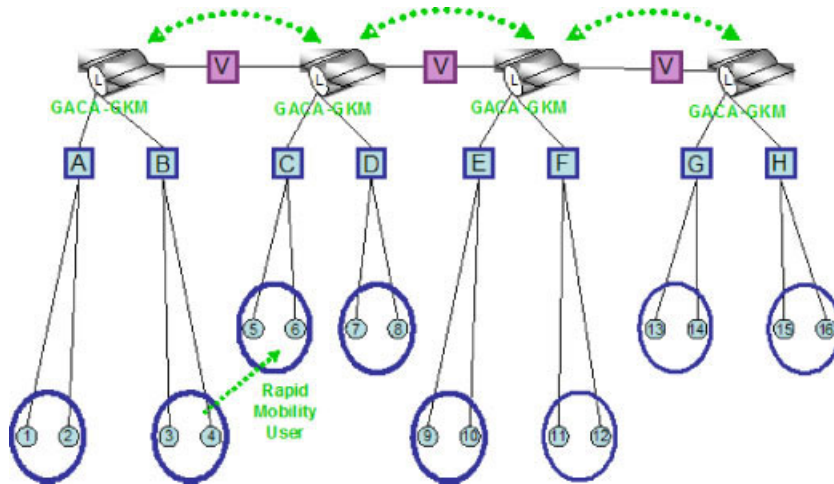


Fig. 1. Hubenko architecture.

increases system scalability for secure multicast in a low earth orbit (LEO) satellite environment [11]. Figure 1 shows a conceptual model of the Hubenko architecture in a LEO satellite environment. The LEO satellites are represented by satellite figures, group keys are represented by letters, and users are represented by numbers. In this architecture, LEO satellites form a cluster at the top of the hierarchy with a group key “V”. Group Cluster Keys A through H are assigned to a cluster of users. A different group key for each cluster. The satellite application in [11] is essentially a two-tier architecture. Since the group cluster keys are maintained onboard the satellite. The Hubenko architecture is modular in its design. As a result, the underlying multicast routing and rekeying protocol are transparent and can be selected to best fit each unique application.

One of the key features incorporated into the Hubenko architecture is clustering. Clustering divides a single large multicast group into subgroups to reduce the overhead involved when group members join or leave the group. The best features of two clustering architectures, known as Spatial Clustering [12] and Iolus [13] are combined to form the basic framework of the Hubenko architecture. Multicast groups are divided into subgroups (clusters) based on the physical location of its members. By using spatial boundaries the key distribution scheme can exploit the parallelism inherent in different parts of a multicast tree to greatly enhance performance [12]. Using the Iolus framework, all of the clusters are independent and each cluster has its own group leader and secret group key. As a result, if a new member joins or leaves the multicast group, only the affected

cluster needs to rekey as opposed to the entire multicast group. Each cluster is managed by a group security agent (GSA), known as a cluster leader. The cluster leaders work with other cluster leaders to bridge the local multicast traffic from each cluster into all of the other clusters as needed [13]. At the head of the hierarchy is a group security controller that manages all of the cluster leaders and the overall security of the group. This is the job of the satellites in the Hubenko architecture. The number and size of the clusters as well as the number of levels in the hierarchy is flexible depending on the application.

To further increase system scalability, the Hubenko architecture incorporates many of the crucial features of another security architecture known as Gothic [10]. Gothic is a comprehensive architecture that provides GAC. The architecture contains a group policy management system and a group member authorization system. The group policy management system has a group owner who provides a list of authorized members and other appropriate security policies for the group to the access control server (ACS). The group member authorization system provides the core control of the architecture by controlling access to the group [10]. These features strengthen system security by preventing unauthorized users from attempting malicious acts such as traffic analysis or denial of service attacks. The designers of Gothic created the architecture with low computation overhead at the routers, low message overhead, and low support infrastructure requirements [10]. These attributes are ideal for resource-constrained MANETs and, in particular, UAV swarms.

Another important feature of Gothic used in the Hubenko architecture is the group access control aware

group key management (GACA-GKM) [10]. This feature leverages the trust built into the GAC system to reduce the requirements of group key management and obtain substantial overhead reductions in a way that enhances scalability and improves performance in terms of less rekeying overhead [11]. For example, in a typical group key management system, whenever a user joins or leaves a multicast group, the entire system is rekeyed based on the assumption that the new user could have gained access to either the old encrypted data prior to arrival or to new encrypted data after departure. By leveraging the services of the GAC system to ensure no unwanted users have access to the data prior to their validated join or after their departure, a rekey is not required [11].

4. Approach

Although the Hubenko architecture has been designed for a LEO satellite system, this research investigates the feasibility of using the architecture to provide a secure, scalable multicast architecture for UAV swarms in the global information grid (GIG). The Hubenko architecture applied to a UAV swarm is shown in Figure 2. A large UAV such as a Global Hawk has a similar role as the LEO satellite in the original Hubenko architecture; however, this research studies the impact of using the Hubenko architecture in a three-tier hierarchical network with the three layers being a Global Hawk, cluster leaders, and users (UAVs). The Global Hawk is the group security controller, group ACS, and is responsible for the overall security of the entire swarm. “GK” is the multicast group key shared among the cluster leaders and the Global Hawk. Each cluster has its own cluster key represented by “CK n ”. The black lines represent communication links while the circles with thick lines represent cluster boundaries. The dashed lines repre-

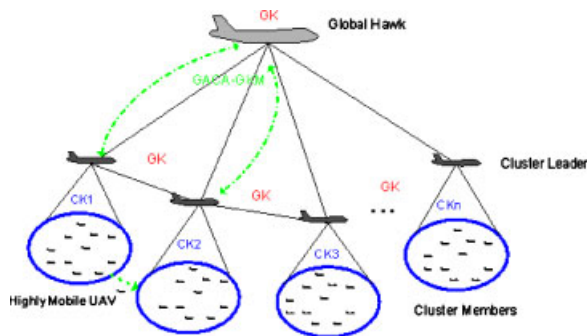


Fig. 2. Hubenko architecture applied to UAV swarm.

sent the GACA-GKM on the Global Hawk, which communicates with the cluster leaders to manage access to the group. Instead of satellite spot beams dictating the number and size of the clusters, this is constrained by the capabilities of the UAVs selected as cluster leaders. When a multicast group first forms, the Global Hawk assigns UAVs as either cluster leaders or cluster members based on their capabilities and location. Ideally medium-sized UAVs are assigned as cluster leaders since they have greater range, endurance, and processing capabilities. The cluster leaders communicate with the Global Hawk flying at an altitude of about 15 km and all of the UAVs in their respective clusters. To increase available bandwidth and avoid transmission collisions the cluster leaders loiter above their clusters and use directional antennas aimed at their cluster. The cluster leaders communicate amongst each other to keep their clusters from overlapping.

The other architectures evaluated in this study are the baseline and the cluster. The baseline architecture for a swarm of UAVs is a flat model, consisting of the swarm and the multicast group leader, which is the Global Hawk. It includes the basic security functions of key generation, key storage, key agreement, and group key distribution to provide a dynamic application proof-of-concept [11]. The entire swarm shares a single SEK and thus every swarm member is rekeyed on a member join or departure. The cluster architecture is an enhanced baseline architecture that includes the clustering concepts from Spatial Clustering and Iolus. Each cluster is independent and has its own unique SEK. As a result, each cluster only needs to be rekeyed when there is a join to, or departure from its cluster.

5. System Description

The System Under Test (SUT) is the UAV Swarm Group Communication System. It consists of the security architecture, wireless network, UAVs, and the multicast routing protocol. The component under test (CUT) is the security architecture. Specifically, the Hubenko architecture is compared to a baseline flat architecture and a basic clustered architecture.

The workload of the SUT is ultimately the amount of multicast traffic that needs to be distributed. This study specifically focuses on reducing the traffic and overhead associated with group key management and distribution. Thus, the amount of multicast traffic related to group key management depends upon several parameters including the size of the swarm

(multicast group), the number and rate of joins and departures to the multicast group, the swarm's mobility, the number of clusters, and the length of the group key. The workload to the SUT is generated by varying these parameters. For example, increasing the swarm size, the group join rate, the group departure rate, the swarm's mobility, and the length of the key will all increase the amount of rekey operations necessary to secure the swarm. As a result the overall amount of multicast traffic increases, thus increasing the workload to the SUT. On the other hand, decreasing the swarm size, the group join rate, the group departure rate, the swarm's mobility, and the length of the key will decrease the amount of rekey operations necessary, thus reducing multicast traffic and the workload to the SUT. The system parameters consist of the transmission range, bandwidth, the physical layer and MAC standard, battery power, processing capabilities, cluster diameter, and UAV speed.

6. Experimental Design

The work by Hubenko in [11] provides insight into the impact of the multicast group size and mobility on each of the investigated architectures. However, the activity and characteristics of the multicast groups modeled in that work do not reflect a realistic scenario for a swarm of UAVs. Hubenko's study models a multicast group whose members join within a fixed time, with some of the members leaving after random intervals. This is visually represented by Scenario 1 in Figure 3. There may be some applications when this model will properly characterize a UAV swarm, but the multicast activity represented by Scenario 2 in Figure 3 is a better model of a UAV swarm's activity. Scenario 2 represents a multicast group with continuous departures, and rejoins to the group. Most of the envisioned missions of UAV swarms (continuous border patrol, battlespace surveillance, ISR etc.) require the swarm sustains itself for prolonged periods of time. This could be several hours or even several

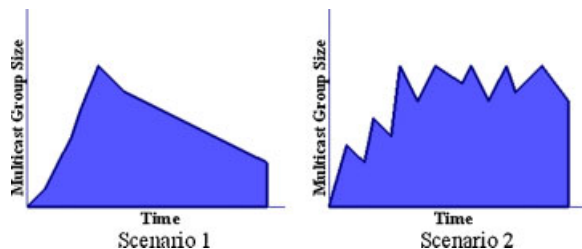


Fig. 3. Multicast group activity for the tested scenarios.

days. Currently small UAVs (SUAVs), which comprise the bulk of the swarm, have limited battery life typically ranging from 1 to 3 h [14]. Therefore, in order for the swarm to sustain its strength and size, its members will need to depart and rejoin several times throughout the duration of the mission to replace or recharge batteries.

Thus, this study tests the Hubenko architecture using Scenario 2 in addition to Scenario 1, to represent different mission requirements placed on UAV swarms. The scenarios are distinguished by the multicast group activity over the simulation period. Scenario 1 represents the scenario where UAVs join the swarm and must depart after their batteries are depleted. None of the departing UAVs rejoin the group. In Scenario 2, the UAV swarm joins the multicast group, but there are continuous departures and rejoins over a longer period. The burden of continuous departures and rejoins to the multicast group will fully test the architectures for a UAV swarm.

6.1. Performance Metrics

As group key management is one of the most complex and resource-intensive operations on the network, the performance metrics should measure how efficient and scalable the security architecture is in terms of group key management. Thus, the following performance metrics are defined:

- *Total Keys*: The total number of keys distributed during the simulation period.
- *Average Rekeys*: Average number of times a UAV must rekey during the simulation period.

Similar metrics were used to evaluate the performance of the Hubenko architecture in the LEO satellite environment as well as related work in the area of secure group communications [11]. These metrics are also relevant in determining potential security performance improvements [15,16].

In addition to the metrics listed above, Scenario 2 also measures:

- *Average Bandwidth*: The average amount of bandwidth used to rekey for a group rekey operation.
- *Battery Consumed*: The average percentage of battery consumed by a UAV to rekey during the simulation period.

These metrics are very important in an environment such as an autonomous UAV swarm where battery

capacity and bandwidth are often limited and costly. These metrics also further highlight the cost associated with rekeying.

In order to calculate *average bandwidth* and *battery consumed*, a few assumptions are made to simplify the experiments. An encryption key length of 256 bits is chosen, as it is a standard key length in the popular Advanced Encryption Standard (AES). Thus, the size of the network layer packet used to distribute the SEK on a rekey is 688 bits as shown below:

$$\begin{aligned} \text{Packet Size} &= \text{MAC Header} + \text{CRC} \\ &\quad + \text{Encryption Key} + \text{IP Header} \\ &= 240 \text{ bits} + 32 \text{ bits} + 256 \text{ bits} + 160 \text{ bits} \\ &= 688 \text{ bits} \end{aligned}$$

The *average bandwidth* is calculated by summing all the rekeys for each UAV performed over the simulation period multiplied by the packet size and divided by the number of seconds in the simulation. This yields the average bits per second (bps). This calculation assumes that each UAV is rekeyed directly by its cluster leader or by the Global Hawk in the case of the baseline architecture. Also, this calculation only takes the packet with the SEK into account. Management or acknowledgement packets are not used in the calculation because they depend on the specific higher level protocols used. This calculation also assumes a pairwise rekey between the cluster leader (Global Hawk) and each UAV, which results in one separate message for each UAV (n messages).

The same assumptions used to calculate *average bandwidth* are also applied to calculate *battery consumed*. In addition, assumptions about the battery and radio are necessary. The representative battery chosen for the simulations is the Thunder Power Lithium Poly battery, which has a usable voltage range from 14.0 to 16.7 V and a 4,200 mA-hr capacity [17]. This battery is currently being used to power UAVs for swarming applications [17]. The representative radio chosen for the simulations is the Ubiquiti Networks Super-Range9 radio, which is also currently being used in conjunction with the selected battery in UAV research [17]. The SuperRange9 is a 900 MHz wireless radio, which features up to 700 mW of output power, -88 dBm receive sensitivity performance (for the 11 Mbps data rate), and has proven non-line-of sight distances over 20 km [18]. Transmit and receive are 1200 mA and 500 mA respectively [18]. The range and capabilities of the selected radio and battery make

Table I. Energy consumption symbols.

E_{Rx}	Energy consumed from receiving (mA-hr)
E_{Tx}	Energy consumed from transmitting (mA-hr)
b_T	Bits transmitted
b_R	Bits received
d_T	Current draw from transmitting (mA-hr)
d_R	Current draw from receiving (mA-hr)
r	Data rate (bits/second)

the assumed communication ranges for the three architectures viable.

With the battery and radio selected, there is enough information to calculate *battery consumed*. First, the energy consumed to rekey is found, which consists of the energy consumed to transmit the rekey packet and the energy consumed to receive the rekey packet. The equations used to calculate the energy consumed to receive and transmit are shown in (1) and (2) respectively [19]. The symbols used in the equations are defined in Table I. The bits transmitted and received are the number of bits in the rekey packet (688). The current draw from transmitting and receiving are taken from the radio's datasheet, and the data rate is assumed to be 11 Mbps.

$$\begin{aligned} E_{Rx} &= \frac{b_R \times d_R}{\left(3600 \frac{s}{hr}\right) r} \\ &= \frac{688 \times 500 \text{ mA}}{\left(3600 \frac{s}{hr}\right) 11 \text{ Mbps}} = 0.0000087 \text{ mA-hr} \end{aligned} \quad (1)$$

$$\begin{aligned} E_{Tx} &= \frac{b_T \times d_T}{\left(3600 \frac{s}{hr}\right) r} \\ &= \frac{688 \times 1200 \text{ mA}}{\left(3600 \frac{s}{hr}\right) 11 \text{ Mbps}} = 0.0000208 \text{ mA-hr} \end{aligned} \quad (2)$$

The results of Equations (1) and (2) are divided by the battery capacity to get a percentage of battery consumed to receive a rekey packet and transmit a rekey packet.

$$\begin{aligned} \text{Battery Consumed Rx} &= \frac{0.0000087 \text{ mA-hr}}{4200 \text{ mA}} \\ &\quad \times 100 = 0.00000207\% \end{aligned}$$

$$\begin{aligned} \text{Battery Consumed Tx} &= \frac{0.0000208 \text{ mA-hr}}{4200 \text{ mA}} \\ &\quad \times 100 = 0.00000495\% \end{aligned}$$

Table II. Factor levels for scenario 1.

Factor	Level 1	Level 2	Level 3	Level 4
Swarm size	40	100	200	500
Swarm mobility	25%	75%		
Join rate	15%	30%		
Departure rate	25%	75%		
Architecture	Baseline	Cluster	Hubenko	

These equations are used in the simulation to calculate an overall average percentage of battery consumed by a UAV to rekey during the simulation period.

6.2. Experimental Factors and Parameters

Tables II and III summarize the factors selected from the system and workload parameters for Scenarios 1 and 2 respectively. These factors are varied to determine the impact they have on the security performance of the UAV swarm in terms of the metrics described above.

The following further describes the factors and defines the levels chosen:

- *Swarm Size:* The number of UAVs in the swarm impacts the total number of keys to be distributed and also increases the overall activity of the swarm, thereby increasing the number of times a UAV needs to rekey. Based on proposed UAV swarms and possible applications the levels selected for Scenario 1 are 40, 100, 200, and 500 UAVs. Scenario 2 also includes 1,000 UAVs to further increase the workload.
- *Swarm Mobility:* This is the percentage of the swarm that is highly mobile. In this study, UAVs are defined as highly mobile if they travel outside of a 5 km radius, whereas UAVs that stay within a 5 km radius are defined as loiterers. A highly mobile environment requires much more rekeying overhead than one in which UAVs loiter in the same general area for long periods of time. The levels

Table III. Factor levels for Scenario 2.

Factor	Level 1	Level 2	Level 3	Level 4	Level 5
Swarm size	40	100	200	500	1,000
Swarm mobility	25%	50%	75%	90%	
Architecture	Baseline	Cluster	Hubenko		

selected for Scenario 1 are 25 and 75%. In addition to these levels, Scenario 2 includes 50 and 90% swarm mobility levels.

- *Group Join Rate:* This is the percentage of the simulation time it takes for the entire swarm to initially join the multicast group. The rate at which UAVs join the multicast group has an impact on the overhead necessary to maintain overall security of the swarm. The levels chosen are 15 and 30%. Thus, when the rate is set to 15%, there will be several more joins to the multicast group in a shorter amount of time compared to when the rate is set to 30%. The group join rate for Scenario 2 is fixed.
- *Group Departure Rate:* This is the percentage of the swarm that departs the multicast group prior to the end of the simulation. The number of departures from the multicast group impacts the overhead necessary to maintain overall security of the swarm. The levels chosen for Scenario 1 are 25 and 75%. The UAVs that depart the group do so after a normally distributed amount of time. The group departure rate for Scenario 2 is not a factor because it is set to 100% for all of the simulations.
- *Security Architecture:* This is the CUT. The security architecture impacts the total number of rekeying operations and the overall security performance of the system. The levels selected are the baseline (flat architecture), cluster, and Hubenko.

The parameters of the system are the properties, which when changed can impact the performance of the system. The fixed experimental parameters are displayed in Table IV and further described here:

- *PHY/MAC Standard:* The physical layer and media access control standards define channel access and data encoding, modulation, and transmission. IEEE

Table IV. Fixed parameter values.

Parameter	Scenario 1 value	Scenario 2 value
PHY/MAC standard	IEEE 802.11b	IEEE 802.11b
Bandwidth	11 Mbps	11 Mbps
Processor speed	1.8 GHz	1.8 GHz
UAV speed	25 m/s	25 m/s
Battery capacity	4,200 mA-hr	4,200 mA-hr
Group key length	256 bits	256 bits
Number of clusters	10	10
Cluster diameter	10 km	10 km
Simulation length	7,200 time steps (2 h)	43,200 time steps (12 h)

- 802.11b is a widely known technology and the current standard of choice for similar research [20].
- *Bandwidth:* The channel bandwidth restricts how much data can be transmitted to the swarm per second. IEEE 802.11b has a maximum bandwidth of 11 Mbps.
 - *Processing Capabilities:* This affects the ability of the UAVs to generate keys and perform encryption and decryption operations. UAVs used in similar research are currently equipped with a Kontron 1.8 GHz processor with 1 GB memory [17].
 - *UAV Speed:* UAV speed impacts how fast and to what degree the network topology changes. A reasonable speed given the expected size and maneuverability of a typical UAV in HARVEST is 25 m/s [21].
 - *Battery Capacity:* This affects the ability of the UAVs to transmit and receive rekey packets. UAVs used in similar research are currently equipped with a Thunder Power Lithium Poly battery (TP4200-4S2PB) with a 4,200 mA-hr capacity [17].
 - *Group Key Length:* This affects the security of the system and the size of the rekey packets. Larger keys increase the security of the system, but require more bandwidth, processing power, and storage. This study assumes a key length of 256 bits, which is a standard length for AES encryption.
 - *Number of Clusters:* This impacts the scalability, efficiency, and communication overhead required in the cluster and Hubenko architectures. The ideal number of clusters varies depending on the situation and may be constrained by resources. Since cluster analysis is beyond the scope of this research, the number of clusters for this study is set at 10 to allow for comparison to previous work [11].
 - *Cluster Diameter:* This is a function of the antenna, transmission range, and altitude of the UAV chosen as the cluster leader and affects the swarm's coverage area. Based on the current capabilities of medium-sized UAVs the cluster diameter is chosen to be 10 km.
 - *Simulation Length:* Longer simulations have more activity such as joins and departures and more mobility among the clusters. The simulation length for Scenario 1 is 2 h which is near the end of the endurance of smaller UAVs [14]. The simulation length for Scenario 2 is 12 h. This represents UAVs having the ability to swap out batteries and rejoin the swarm after a certain amount of time.

6.3. Experimental Setup

The experimental setup for this study consists of two sub-experiments, each with a full-factorial design

with the factors listed in Tables I and II. The first sub-experiment simulates Scenario 1 and closely resembles the experiments in [11], allowing for comparisons. It consists of 8 repetitions for each configuration, requiring a total of 768 simulation runs ($4 \times 2 \times 2 \times 2 \times 3 \times 8 = 768$). The second sub-experiment simulates Scenario 2, which is a new test of the Hubenko architecture. It consists of 20 repetitions for each configuration, requiring a total of 1,200 runs ($5 \times 4 \times 3 \times 20 = 1,200$). Thus, the overall experiment will consist of 1,968 simulation runs. The number of repetitions provides a narrow enough confidence interval while minimizing the number of experiments necessary. Each of the repetitions for the same configuration use a different seed for the random number generator which affects the various aspects of the simulation including the join time, departure time, assigned cluster, and mobility of the each UAV.

7. Simulation Environment

Currently a swarm of autonomous unmanned vehicles is still in the concept stage and an actual system is not yet fielded. Thus, measurement of an actual system is not feasible for this study. In addition, using an actual system, if one existed, would be very costly and time consuming. Using an analytical model is also not a viable option because there is no such model that can be adapted to this scenario. Thus, the best evaluation technique for this study is a simulation. Because this study is specifically concerned with reducing security overhead in the form of group key management, much of the details about data transmission, packets, and routing can be abstracted away. This makes MATLAB the best choice to perform the simulation for this study.

A discrete event computer simulation using MATLAB, (version R2007a) is developed to evaluate the performance of the baseline, cluster, and Hubenko architectures in terms of group key management and distribution in a swarm of UAVs. The simulation environment is a modified version of the one used in [11], which models a satellite-based multicast network. However, several modifications to the simulation are made to characterize a swarm of UAVs and this study's experimental design. Although a detailed description of the original simulation environment can be found in [11], several significant modifications are described below.

In the original simulation the time steps are left undefined, however for the purpose of this research

one time step represents one second. This means if a UAV joins the multicast group at the beginning of the one second interval, it will not receive a multicast key until the end of the interval, thus having to wait up to one second to start receiving multicast data. The same logic applies to a UAV leaving the multicast group. If a UAV leaves the multicast group at the beginning of the one second interval, it still may be able to receive multicast data for up to one second because the rest of the UAVs in the multicast group will be rekeyed at the end of the one second interval. In actual use applications larger or smaller intervals can be used depending on the security needs of the system.

Another important modification is how the metrics *total keys* and *average rekeys* are calculated for the baseline study. Because the original study deals with a geographically-widespread satellite environment, the baseline architecture requires a rekey operation anytime a user moved from one spot beam to another regardless if the is was already a member of the multicast group. However, in this study the baseline architecture is a large UAV acting as the single multicast group leader with a swarm of smaller UAVs locally distributed out within its range. Because it is assumed that the multicast group leader can directly and/or indirectly transmit to all members of the swarm, there is no need to rekey as swarm members move within that range. For example, the highly mobile UAV in Figure 2 would not cause a rekey in the baseline study because clusters are non-existent and Global Hawk acts as the multicast group leader for the entire swarm.

The simulation environment is also modified to simulate both scenarios. The original study only simulates the multicast group activity of Scenario 1 shown in Figure 3. Aside from the changes mentioned above both scenarios require changes to the experimental parameters and factors to correspond to the experimental design and properly model a UAV swarm.

7.1. Scenario 1 Simulation

In Scenario 1, 1.2h or 7,200 discrete time steps of rekeying activity in a UAV swarm is simulated. During each simulation run, all factors (join rate, departure rate, mobility rate, and swarm size) are held constant, but the three architectures are tested under the same conditions. Each UAV is randomly assigned an initial join time to the multicast group, an initial cluster, a mobility type (highly mobile or loitering), and a departure time (if applicable). All

of the random assignments are based on a uniform distribution. The join rate determines whether the UAVs randomly join within the first 15 or 30% of the simulation time. The departure rate determines the percentage of the swarm that departs the group before the end of the simulation (either 25 or 75%). The mobility rate determines the percentage of the swarm assigned as highly mobile or loiterers. The UAVs assigned as highly mobile change clusters throughout the simulation based on their velocity of 25 m/s, while the UAVs assigned as loiterers remain in their initial assigned cluster. The *total keys* and *average rekeys* are tracked for each individual UAV for each of the three tested architectures.

7.2. Scenario 2 Simulation

In Scenario 2, 12h or 43,200 discrete time steps of rekeying activity in a UAV swarm is simulated. This scenario allows UAVs to rejoin the multicast group after departing and models the situation where a UAV swarm needs to be sustained for a long period of time, longer than a UAV's typical battery life. Thus UAVs depart the swarm to recharge or exchange their batteries and then rejoin the group. The join rate is not a factor and is held constant. The departure rate is also not a factor in this scenario because the swarm members continuously depart and rejoin the multicast group. Similar to Scenario 1, the mobility rate and swarm size are held constant during each run and the three architectures are tested simultaneously under the same conditions.

In the beginning of the simulation, each UAV is randomly assigned an initial join time during the first simulated hour (3,600 time steps). Each UAV is also randomly assigned a duration (battery life) ranging from 30 to 180 min. This represents the battery capacities of the various small UAVs currently in operation as can be found in the DoD's Unmanned Systems Roadmap [14]. Also, varying battery capacities would be typical in a heterogeneous UAV swarm. Each UAV is randomly assigned to an initial cluster and as highly mobile or loitering. After a UAV initially joins the multicast group, it stays for its randomly-assigned duration and then departs. It then rejoins the swarm 30 min later representing the time to swap out its battery. This is repeated throughout the simulation. The *total keys*, *average rekeys*, *average bandwidth*, and *battery consumed* are tracked for each individual UAV for each of the three tested architectures.

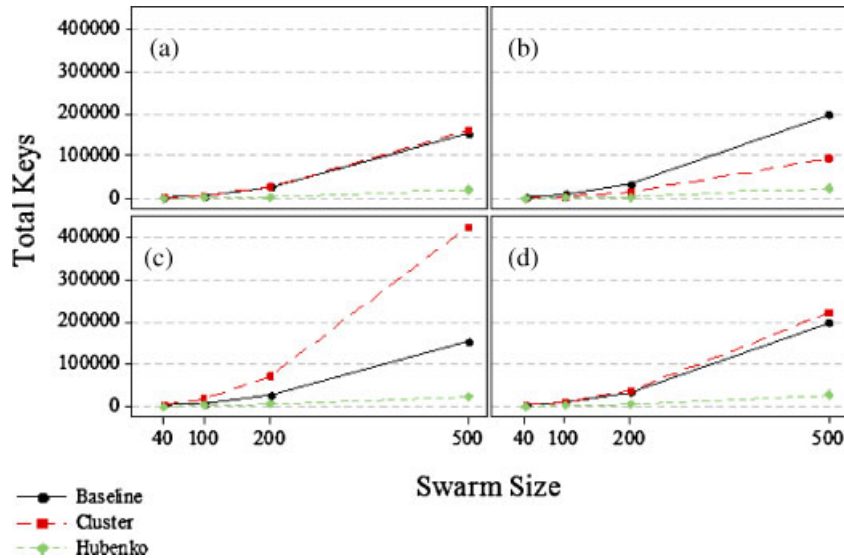


Fig. 4. Total keys *versus* swarm size (Scenario 1); (a) 25% mobile, 25% departure rate (b) 25% mobile, 75% departure rate (c) 75% mobile, 25% departure rate (d) 75% mobile, 75% departure rate.

8. Experimental Results

8.1. Scenario 1

The results from Scenario 1 are displayed in Figures 4 and 5. Both figures contain four plots labeled a–d, representing the different configurations of the factors swarm mobility and departure rate. The factor join rate is not included because it does not significantly impact the variation in either response. The 95% confidence intervals are not shown on the plots because they are too narrow to be distinguished.

Figure 4 displays *total keys versus* swarm size. As expected, more keys are distributed in the system when the swarm size is the largest and mobility is high. Also, as predicted, the fewest keys are distributed in the system with the Hubenko architecture. The baseline and cluster architectures' performance relative to each other vary depending on the swarm's mobility and departure rate. By visual inspection it can be seen that the Hubenko architecture has statistically significant differences compared to the baseline and cluster architectures. Also, statistically significant differences can be seen among the various swarm sizes.

Using pair-wise comparisons of the mean responses at the 0.05 level of significance, each level of the swarm size as well as both levels of mobility have significant statistical differences from all other levels. The Hubenko architecture is statistically different

from the cluster and baseline architectures, but the baseline and cluster architectures are not statistically different from each other. The two levels of both the *departure rate* and the *join rate* are not statistically different. Using the mean response values across all factors, the total keys distributed in the system is 86.2% less in the Hubenko architecture compared to baseline and 89.2% less compared to the cluster architecture.

Average rekeys versus swarm size is shown in Figure 5. Comparing Figures 4 and 5, it can be seen that the factors have similar effects on *average rekeys* as they did on the *total keys*. Also, by visual inspection it can be seen that the Hubenko architecture has significant statistical differences compared to the baseline and cluster architectures and has the lowest *average rekeys* across all factor combinations. Using 95% confidence intervals, significant statistical differences exist among the various swarm sizes and the two mobility levels. Using the mean response values across all factor levels, the *average rekeys* per UAV is 84.9% less in the Hubenko architecture compared to the baseline and 87.1% compared to the cluster architecture.

Several important conclusions can be drawn from Scenario 1. First and foremost, the growth rate of both responses *versus* the swarm size and mobility is significantly smaller using the Hubenko architecture. This demonstrates the architecture is both scalable and efficient as predicted. Also, we learn that the swarm size and architecture cause the biggest variation in

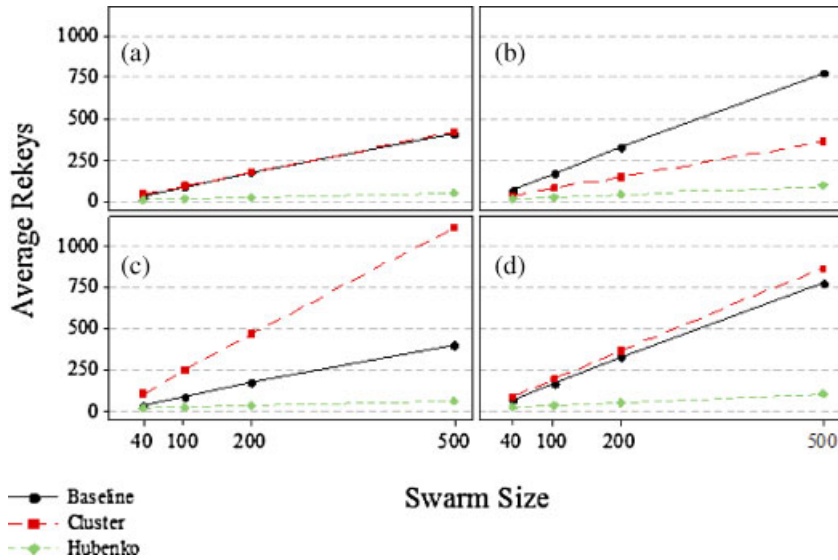


Fig. 5. Average rekeys versus swarm size (Scenario 1); (a) 25% mobile, 25% departure rate (b) 25% mobile, 75% departure rate (c) 75% mobile, 25% departure rate (d) 75% mobile, 75% departure rate.

both responses followed by the swarm’s mobility. Although the *join rate* has a small *p*-value using analysis of variance (ANOVA), it causes less than 1% of the variation in both responses. Therefore, join rate is dropped as a factor in Scenario 2.

8.2. Scenario 2

The results from Scenario 2 are displayed in Figures 6–9. Each figure contains four plots labeled

a–d, which correspond to different mobility levels: 25, 50, 75, and 90%. The 95% confidence intervals are not shown on the plots because they are too narrow to be distinguished.

Total keys versus swarm size is shown in Figure 6. By visual inspection it can be seen that a larger the swarm size and higher mobility increases the total keys distributed, while using the Hubenko Architecture decreases the total keys distributed. Unlike Scenario 1, the cluster architecture outperforms the

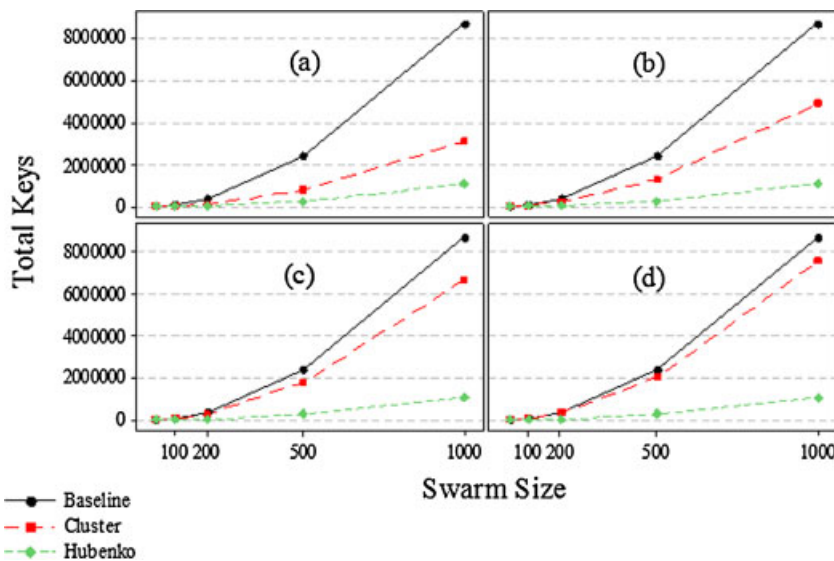


Fig. 6. Total keys versus swarm size (Scenario 2); (a) 25% mobile (b) 50% mobile (c) 75% mobile (d) 90% mobile.

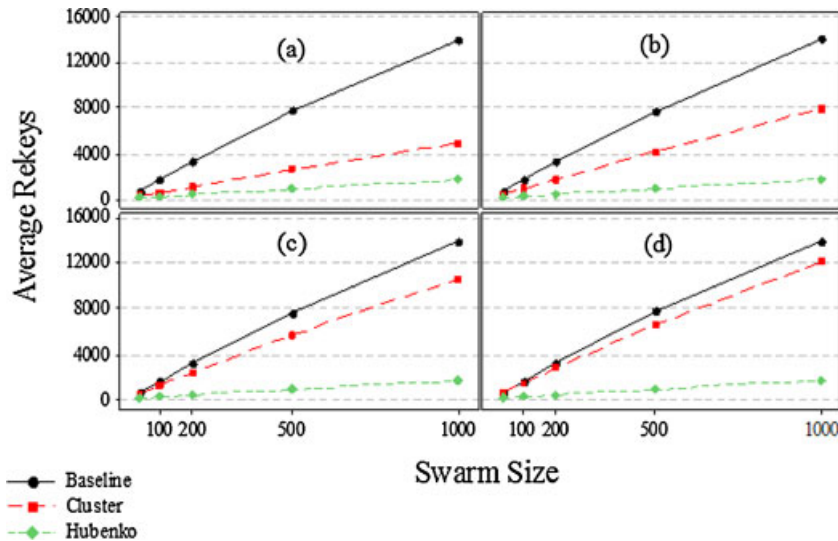


Fig. 7. Average rekeys versus swarm size (Scenario 2); (a) 25% mobile (b) 50% mobile (c) 75% mobile (d) 90% mobile.

baseline architecture in every situation which reveals the negative impact departures and rejoins have on the baseline architecture. Using pair-wise comparisons of the mean responses at the 0.05 level of significance, each level of the swarm size has significant statistical differences from all other levels. Each architecture is also significantly different from all other architectures. Using the mean response values across all factors, 87.6% less keys are distributed in the Hubenko architecture compared to the baseline and 85.0% less keys are distributed compared to the cluster architecture.

Figure 7 displays the *average rekeys versus* the swarm size. Statistically significant differences can be seen among the three architectures and the various swarm sizes. Using pair-wise comparisons of the mean responses at the 0.05 level of significance, each level of the swarm size has significant statistical differences from all other levels. Each architecture is also statistically different from all other architectures. The 25% mobility level and the 90% mobility level are the only mobility levels with significant statistical differences. A UAV in the Hubenko architecture rekeys an average of 87.3% less than a UAV in the baseline architecture. Similarly, a UAV in the Hubenko architecture rekeys an average of 79.9% less than a UAV in the cluster architecture.

Figure 8 displays *average bandwidth versus* the swarm size. In terms of reducing the use of limited resources, such as bandwidth, the power of the Hubenko architecture is evident. At the 25% mobility

level, both the cluster and Hubenko architecture scale well, relative to the baseline, as the swarm size increases. However, once mobility increases, the bandwidth used by the cluster architecture nears that of the baseline, while the Hubenko architecture is minimally affected. At the 90% mobility level, the Hubenko architecture uses an average of 85.3% less bandwidth than the cluster architecture and 87.3% less than the baseline architecture.

Figure 9 displays *battery consumed versus* the swarm size. Interestingly, the baseline architecture outperforms the cluster architecture in terms of the response. In the baseline architecture, the Global Hawk uses fuel, not batteries and rekeys all of the swarm members. Thus, battery is only consumed when a swarm member receives a new key. However, in the cluster and Hubenko architectures, the keys are distributed by cluster leaders, which are swarm members themselves, and thus energy is consumed to both transmit and receive a key. Although the results appear insignificant as the percentage of battery consumed is so small, the relative performance differences among the architectures are very significant. Not included in the simulation are routing, lost packets, and higher-level protocols that add in reliability. Thus, the simplest case is assumed to rekey the swarm: one packet transmitted to, and received by each swarm member containing the key. When routing and reliable protocols are factored into future experiments, the percentage of battery consumed to rekey will undoubtedly increase. Thus, the rate at

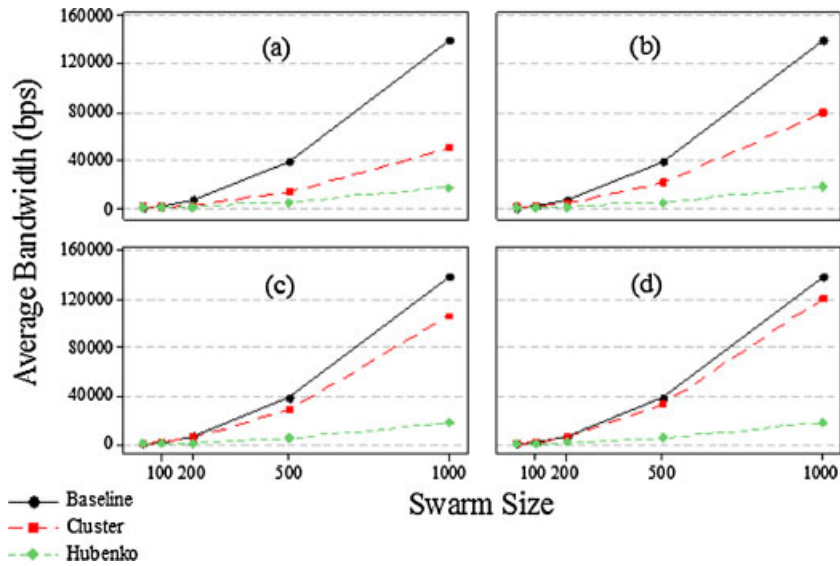


Fig. 8. Average bandwidth *versus* swarm size (Scenario 2); (a) 25% mobile (b) 50% mobile (c) 75% mobile (d) 90% mobile.

which the percentage of battery consumed increases with swarm size and mobility provides more useful information. Figure 9 shows the growth rate of the response *versus* swarm size and mobility is the lowest in the Hubenko architecture.

8.3. Overall Analysis

Several conclusions can be drawn from the simulations conducted. Most importantly, statistical analysis of the

data confirms the hypothesis. The Hubenko architecture provides statistically significant performance gains over commonly used baseline and cluster group communication security architectures. By taking advantage of spatial clustering to decrease the negative performance impact of joins and departures, and integrating GACA-GKM to decrease the negative performance impact of highly mobile UAVs, the Hubenko architecture outperforms the baseline and cluster architectures in all of the conducted experiments. Using the data

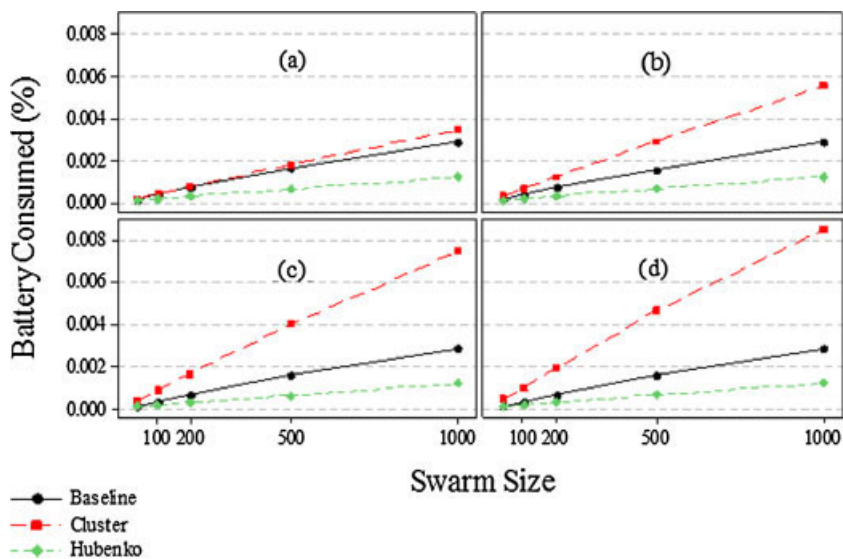


Fig. 9. Battery consumed *versus* swarm size (Scenario 2); (a) 25% mobile (b) 50% mobile (c) 75% mobile (d) 90% mobile.

from both scenarios, the following summarizes the performance gains achieved by the Hubenko architecture compared to the baseline architecture (ranging from the smallest to the highest gains across all configurations):

- 57.8–87.6% less total keys distributed
- 59.6–87.9% less rekeys per UAV
- 73.0–87.9% less bandwidth used to rekey
- 16.9–58.8% less battery consumed to rekey

Similarly, the following summarizes the performance gains achieved by the Hubenko architecture compared to the cluster architecture (ranging from the smallest to the highest gains across all configurations):

- 55.2–94.9% less total keys distributed
- 59.0–94.8% less rekeys per UAV
- 55.2–85.4% less bandwidth used to rekey
- 54.3–85.4% less battery consumed to rekey

Also important to realize these performance gains also coincide with an overall improvement in the security of the system via group access controls and independent SEKs for each cluster.

Other conclusions that can be drawn from the overall analysis of the simulations are the significance and effects of the factors. First, comparing data from the two scenarios, it can be seen that the longer simulation time, and the ability of UAVs to continuously depart and rejoin the swarm significantly increases *total keys* and *average rekeys*. As expected, the swarm size significantly contributes to the variation in all of the responses, causing the most variation in all but one of the measured responses. The architecture is the second largest contributing factor in all but one of the responses, where it is the largest. As discussed previously, the join rate is significant according to the *p*-value from the general linear model, but it contributes very little to the variation in the measured responses. The mobility of the swarm has no effect on the baseline architecture, but has significant effects in both the Hubenko and cluster architectures.

9. Conclusion

The Hubenko architecture can be successfully applied to a swarm of autonomous UAVs. Furthermore, the Hubenko architecture significantly outperforms the two other security architectures studied in terms of

reducing *total keys*, *average rekeys*, *average bandwidth*, and *battery consumed*. By taking advantage of spatial clustering to decrease the negative performance impact of joins and departures, and integrating GACA-GKM to decrease the negative performance impact of highly-mobile UAVs, the Hubenko architecture is a very efficient and scalable architecture ideally suited for a swarm of UAVs.

In most cases, statistical analysis of the metrics finds swarm size to be the largest factor contributing to the variation in the responses, followed by the architecture, and the swarm's mobility. The largest performance gains are seen in large, highly mobile swarms, in which UAVs continuously join and depart the group. In this type of environment the Hubenko architecture reduces *total keys*, *average rekeys*, and *average bandwidth* up to 88% compared to the baseline architecture. *Battery consumed* is reduced up to 59% compared to the baseline.

Acknowledgments

This work was supported in part by the Air Force Communications Agency. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

References

1. Kearney D, Jasuinas M. Managing power amongst a group of networked embedded FPGAs using dynamic reconfiguration and task migration. *Dagstuhl Seminar Proceedings 06141*, 2006.
2. De Morais Cordeiro C, Gossain H, Agrawal D. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network* 2003; **17**(1): 52–29.
3. Sun Z, Howarth MP, Iyengar S, Claverotte L. Networking issues in IP multicast over satellite. *International Journal of Satellite Communications and Networking* 2003; **21**: 489–507.
4. Lazos L, Poovendran R. Power proximity based key management for secure multicast in ad hoc networks. *Wireless Networks* 2006; **13**: 127–148.
5. Yadav V, Balakrishnan SN. Communication control in multiple UAV applications. *AIAA Guidance, Navigation, and Control Conference*, Vol. 3. Keystone: CO, United States, 2006; 1428–1461.
6. Kearney D, Jasuinas M. Managing power amongst a group of networked embedded FPGAs using dynamic reconfiguration and task migration. *Dagstuhl Seminar Proceedings 06141*, 2006.
7. Hyland MT, Mullins BE, Baldwin RO, Temple MA. Simulation-based performance evaluation of mobile ad hoc routing protocols in a swarm of unmanned aerial vehicles. *IEEE International Symposium on Pervasive Computing and Ad Hoc Communications (PCAC-07)*, Niagara Falls, Canada, May 2007.

8. Aye W, Siddiqi MU. Key management for secure multicast over IPv6 wireless networks. *EURASIP Journal on Wireless Communications and Networking* 2006; **2006**(2): 1–12.
9. Bruschi D, Rosti E. Secure multicast in wireless networks of mobile hosts protocols and issues. *Mobile Networks and Applications* 2002; **7**: 503–511.
10. Judge P, Ammar M. Gothic: a group access control architecture for secure multicast and anycast. *Proceedings of the IEEE INFOCOM*, New York City, NY, USA, 2002.
11. Hubenko V Jr., Raines R, Baldwin R, Mullins B, Mills R, Grimaila M. Improving satellite multicast security scalability by reducing re-keying requirements. *IEEE Network* 2007; **21**(4): 51–56.
12. Banerjee S, Bhattacharjee B. Scalable secure group communication over IP multicast. *IEEE Journal on Selected Areas in Communications* 2002; **20**(8): 1511–1527.
13. Mitra S. Iolus: a framework for scalable secure multicasting. *Proceedings of ACM SIGCOMM'97*, Cannes, France, 1997.
14. Office of the Secretary of Defense (OSD), Unmanned Systems Roadmap 2007–2032, December 2007. Available: <http://www.fas.org/irp/program/collect/usroadmap2007.pdf>
15. Howarth MP, Iyengar S, Sun Z, Cruickshank H. Dynamics of key management in secure satellite multicast. *IEEE Journal on Selected Areas in Communications* 2004; **22**(2): 308–319.
16. Rafaei S, Hutchison D. A survey of key management for secure group communication. *ACM Computing Surveys* 2003; **35**(3): 309–329.
17. Gruber S. E-mail interview. 30 Oct 2007.
18. Ubiquiti Networks, Inc. (2007) SuperRange9 datasheet, Retrieved Oct 30, 2007 from <http://ubnt.com/downloads/sr9datasheet.pdf>.
19. Jordt G. Evaluation of energy costs and error performance of range-aware, anchor-free localization algorithms for wireless sensor networks. Masters Thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2006.
20. Pack D. E-mail interview. 16 May 2007.
21. Augeri C, Morris K, Mullins B. HARVEST: A framework and co-simulation for analyzing unmanned aerial vehicle swarms. *Military Communications Conference*, 25 Oct 2006, Vol. 23, 1–7.