

**Instructor:** Dr. James Minseok Kwon  
**COURSE:** 4005 -898-01

# **IMPLEMENTATION OF PIKE PROTOCOL**

---

**PIKE: PEER INTERMEDIARIES FOR KEY  
ESTABLISHMENT IN SENSOR NETWORKS**

**BY:** MIHIR DAFTARI

**SUBMITTED:** AUGUST 21, 2008

# Table of Contents

<b>1. Introduction</b>	<b>3</b>
<b>2. Motivation for PIKE</b>	<b>5</b>
<b>3. PIKE Protocol</b>	<b>5</b>
<b>4. Design</b>	<b>7</b>
<b>5. Possible Improvements</b>	<b>14</b>
<b>6. Conclusion</b>	<b>15</b>
<b>7. References</b>	<b>17</b>

## 1. Introduction

A very large scale deployment of wireless sensor networks is taking place recently. These sensor networks provide a solution to a lot of challenging problems like military sensing, traffic monitoring, monitoring physiological parameters of patients, etc. Secret communication is a very important requirement for most of these applications.

For sensor network security, nodes have to be secured against various kinds of attacks like node capture, denial of service attacks, physical tampering, etc. The sensor nodes are usually limited in energy and computational abilities. Unlike traditional networks, sensor nodes are deployed in easily accessible areas and the nodes interact closely with their physical environments posing unique security problems. Because of these unique characteristics traditional security techniques applied for conventional networks can not be used in wireless sensor networks [3]. Asymmetric cryptography approach such as public-key cryptography (PKC) can achieve authentication, integrity, and confidentiality in traditional wired networks and they work great for traditional networks. But considering the limited power, data processing capacity, and memory storage of sensor nodes, traditional asymmetric-key approach is not desirable. A symmetric-key solution is hence preferable in wireless sensor networks. The simplest solution would be the “master-key” approach wherein the entire network shares the master key. This solution would mean that compromise of a single node will make the entire network unsecured. Another approach is to preload all the nodes in the network with shared unique symmetric key between each pair. This will make the network resilient to node capture. Unfortunately this approach is not scalable and consumes too much memory. In a network with  $n$  nodes, each nodes needs to store  $n-1$  keys. Also addition of more nodes to the network later on will require adding a new symmetric key to all the nodes in the system.

Another approach is bootstrapping keys using a trusted base station. This scheme relies on presence of a resource-rich key distribution center (KDC). Here each node can share a single key with the KDC and set up keys with other nodes through the KDC. This scheme has several drawbacks. First is that it makes the KDC a single point of failure. Secondly, in a large multi-hop system the nodes near the KDC are overloaded with messages causing power consumption issues.

A recent technique to establish keys in wireless sensor network without using any centralized mechanism is called random key predistribution schemes. In this scheme a large pool of symmetric keys is chosen and a random subset is distributed to each sensor node. Two nodes that want to communicate search their pool to determine a common key. This scheme is a result of security vs. efficiency tradeoff between the master-key approach and a fully symmetric approach where each node shares a unique key with every other node in the system. As a result it has both advantages and disadvantages from both the schemes.

The first tradeoff we see is giving up security of fully symmetric key distribution to achieve better memory usage of master key approach. The subset of keys distributed to each key has to be large enough to keep the network connected. The number of keys that each node will carry is less than the keys it would have to store in symmetric key approach but more than the master-key approach. The second tradeoff is giving up resistance against node capture to achieve better memory consumption. The attacker will have to compromise sufficiently large number of nodes to reconstruct the complete pool of keys. This number is more than the number in the master-key approach (one), but less than the number required in fully symmetric approach.

The unpredictable network topology makes the distribution of keys a challenging task. There are many random key predistribution schemes based on probability and graph theory or bivariate polynomial calculations. Each scheme tries to simultaneously achieve security and efficiency requirements of the following table [4].

DESIGN REQUIREMENT OF KEY PRE-DISTRIBUTION SCHEME	
Security Requirement	Authentication
	Secrecy
	Resilience against node capture
	Resistance against node replication
	Compromised node revocation
Efficiency Requirement	Fresh node addition
	Network connectivity
	Maximum supported network size
	Minimum memory storage
	Low computational overhead
	Low communication overhead

Figure 1: Table summarizing design requirement for key pre-distribution scheme

## 2. Motivation

The random key predistribution schemes rely on the fact that random graphs are connected with high probability if average degree of nodes is above a threshold. Enough keys need to be pre-distributed to meet the threshold requirement and hence keep the network connected [1]. The biggest advantage of this scheme is that the communication cost remains constant regardless of network size. But to maintain node resilience as the number of nodes increases in the system the number of keys each node has to maintain increases linearly. Also, performing probabilistic key establishment in sparse sensor networks or in a network where node-density is non-uniform could result in disconnected networks.

Peer Intermediaries for Key Establishment (PIKE) is a deterministic key establishment scheme that uses peer sensor nodes as trusted intermediaries for key establishment. It is designed to address several shortcomings of the existing symmetric-key distribution schemes. PIKE can establish keys between nodes regardless of network topology or node density. This scheme is designed to incur sub-linear overheads in memory per node and focused communication load per node while retaining the property of resilience against the compromise of a fraction of the network.

## 3. PIKE Protocol

PIKE combines basic ideas of random key predistribution and Kerberos like key establishment. It uses trusted intermediaries to establish key between nodes. The entire network is converted into a virtual grid of nodes. Each node in the grid is assigned an ID of the form  $(x, y)$  where  $x, y$  belong to the set  $\{0, 1, 2, \dots, \sqrt{n}-1\}$ . Each node  $(x, y)$  is then loaded with a secret key pairwise-shared with each node in the same column and row in the grid. Each nodes stores  $2(\sqrt{n}-1)$  keys and the total number of unique keys generated is  $n(\sqrt{n}-1)$ . The following figure shows a sample virtual ID space for 100 nodes [1]

00	01	02	03	04	...	09
10	11	12	13	14	...	19
20	21	22	23	24	...	29
30	31	32	33	34	...	39
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.		.
90	91	92	93	94	...	99

Fig 2: Sample virtual ID space for 100 nodes. Each number represents a node ID. Dark and light shaded boxes indicate nodes which share unique keys with nodes 91 and 14 respectively. The keys are pairwise-shared and not common to a row; for example 91 and 01 share a key that is distinct from the key shared between 91 and 11. In the above diagram, either node 11 or node 94 can be used to establish keys between nodes 91 and 14. This grid of node IDs is purely virtual space; there is no correspondence to the actual physical location of the nodes.

Figure 2 shows an example of how the scheme works for a network of 100 nodes. Suppose node A wants to send a message to node B. Node A can choose an intermediary in its row and node B's column or in its column and node B's row. The final choice between these options is made based on heuristic (geographic distance for example) or other routing metric. Once the intermediary C is chosen, A encrypts the new key to be shared with B using the key it shares with C and then sends it to C. C decrypts the key and then re-encrypts it using the key it shares with B, and sends it to B. Finally, B acknowledges the receipt of key to A. The following figure shows the key exchange process.

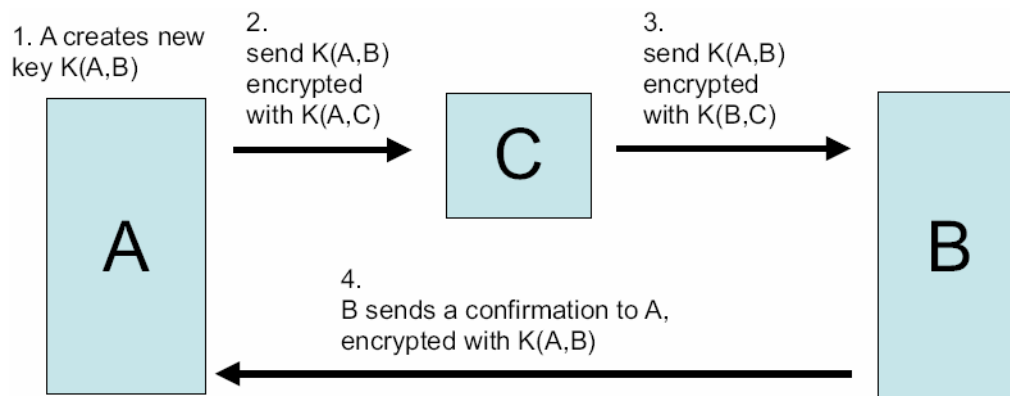


Figure: 3 Key exchange in PIKE

The following figure summarizes the above process.

$$\begin{aligned} A \rightarrow C &: E_{K_{AC}} \{A, B, K_{AB}\}, \text{MAC}_{K_{AC}}(E_{K_{AC}} \{A, B, K_{AB}\}) \\ C \rightarrow B &: E_{K_{BC}} \{A, B, K_{AB}\}, \text{MAC}_{K_{BC}}(E_{K_{BC}} \{A, B, K_{AB}\}) \\ B \rightarrow A &: E_{K_{AB}} \{A, B, N_B\}, \text{MAC}_{K_{AB}}(E_{K_{AB}} \{A, B, N_B\}) \end{aligned}$$

*Figure 4: Estimated Communication Overhead and Security*

## 4. Design and Implementation

### 4.1. Platform used

The Sun SPOT devices are small, wireless, battery powered experimental sensor platforms. They are programmed entirely in Java to allow rapid development of specialized embedded system. The hardware platform includes a range of built-in sensors as well as the ability to easily interface to external devices [4]. The current configuration of the Sun SPOT platform, the eSPOT, has a main processor running the Java VM “Squawk” and which serves as an IEEE 802.15.4 wireless network node. The protocols are implemented on top of the MAC layer. Fig X shows the network stack in the SPOT. It is designed to experiment with wireless routing protocols and many other sensor application including key management applications. The top level of the stack implements protocols similar to TCP and UDP to provide reliable and datagram based applications. The layer underneath is the lowPan layer which handles the packet fragmentation and reassembly and the layers underneath together implement the physical MAC layer.

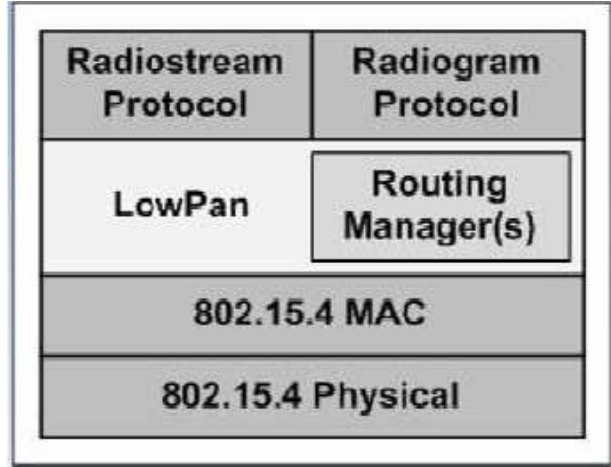


Figure 5: Communication Stack in the Sun SPOTs

## 4.2 Design of Implementation

The implementation of basic PIKE protocol is done at the application layers using the reliable Radiostream Protocol for communication. The application is designed to be multithreaded so that each node can send and receive messages simultaneously. One of the assumptions of the protocol is that nodes are globally addressable in the entire network. The example mentioned for globally addressable communications infrastructure in the paper is GPSR as routing protocol and geographic hash table (GHT) as an address lookup service. In this implementation AODV, which is the default routing protocol in Sun SPOTs, is used as the routing protocol. A GHT or any other kind of global hash table was not implemented since it was beyond the scope of this project. Instead each node had two hard-coded hash tables using which a node could lookup a MAC address for a PIKE ID and a PIKE ID for a MAC address.

There are three kinds of messages designed to achieve key establishment – EstKeyMsg, KeyConfirmMsg, and DataMsg. When a node wants to establish a key for secret communication it first uses the EstKeyMsg to establish key with the destination. The destination node sends a KeyConfirmMsg upon successfully retrieving the key from the source. The DataMsg is used send data once keys are established. The following figures show the message formats. The fields with red border are NOT encrypted as they form the header and read first by the receiver.

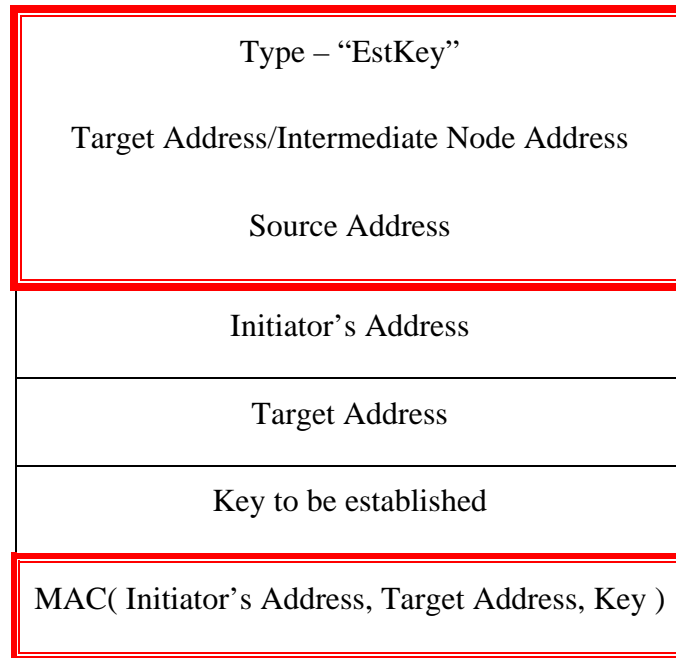


Figure 6: Message format for EstKey Message. The fields with red border form the header of the message and are NOT encrypted whereas the fields with black border form the payload of the message and are encrypted using appropriate pairwise key.

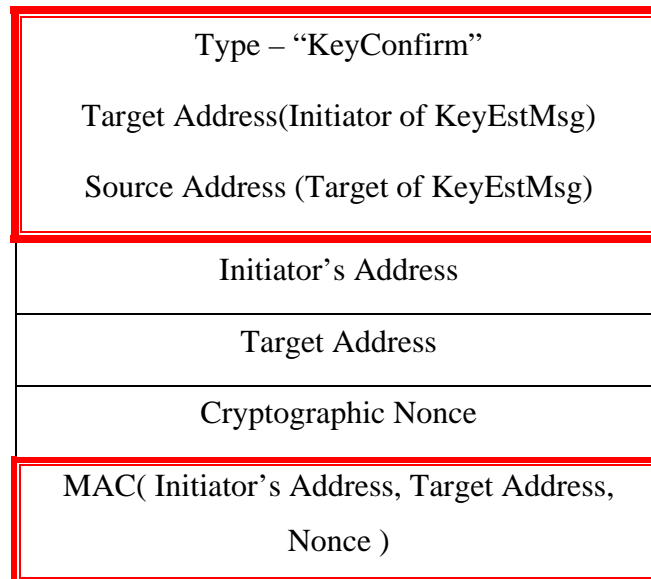
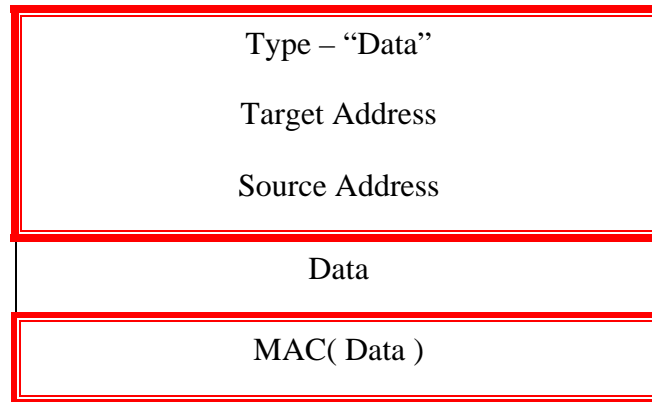


Figure 7: Message format for KeyConfirmMsg. The message format is similar to the KeyEstMsg only difference being that now instead of a key the receiver sends back a cryptographic nonce to acknowledge the receipt of the message.



*Figure 8: Message format of DataMsg. This is the simplest message with just the source, target and encrypted message.*

When a node wants to send a message to another node it first checks to see if there exists a key to encrypt the message with. If it can find the key it will encrypt the message with that key and send it directly to the destination. But if it can not find the key it will find an intermediate node which shares pairwise unique keys with the source node and the destination node. In the paper the author suggests to use some kind of heuristic to determine which peer to choose for key establishment but in the implementation the source node always picks a peer in the same row as itself and the same column as the destination node. For instance, if a node with ID 91 wants to communicate with a node with ID 53, peer with ID 93 will act as an intermediary. The following flowchart describes the entire process a sender thread will execute.

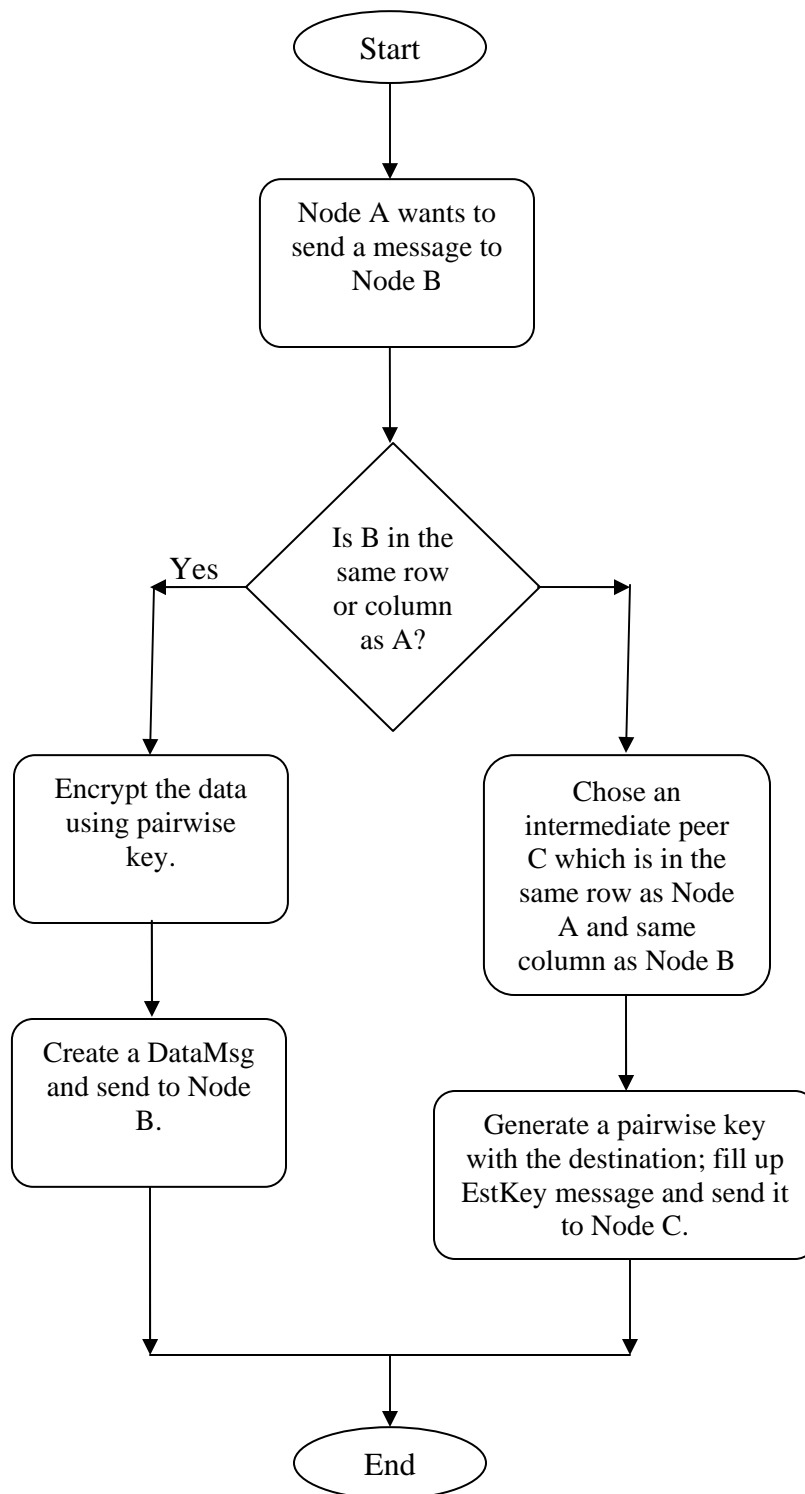


Figure 9: Flowchart of the sender thread.

When a node receives a message it will check and see what kind of message it is and who is the source of the message. It will decrypt the necessary content from the message using the pairwise key and then either forward the message or respond. The following flowchart describes what the processing of a received message.

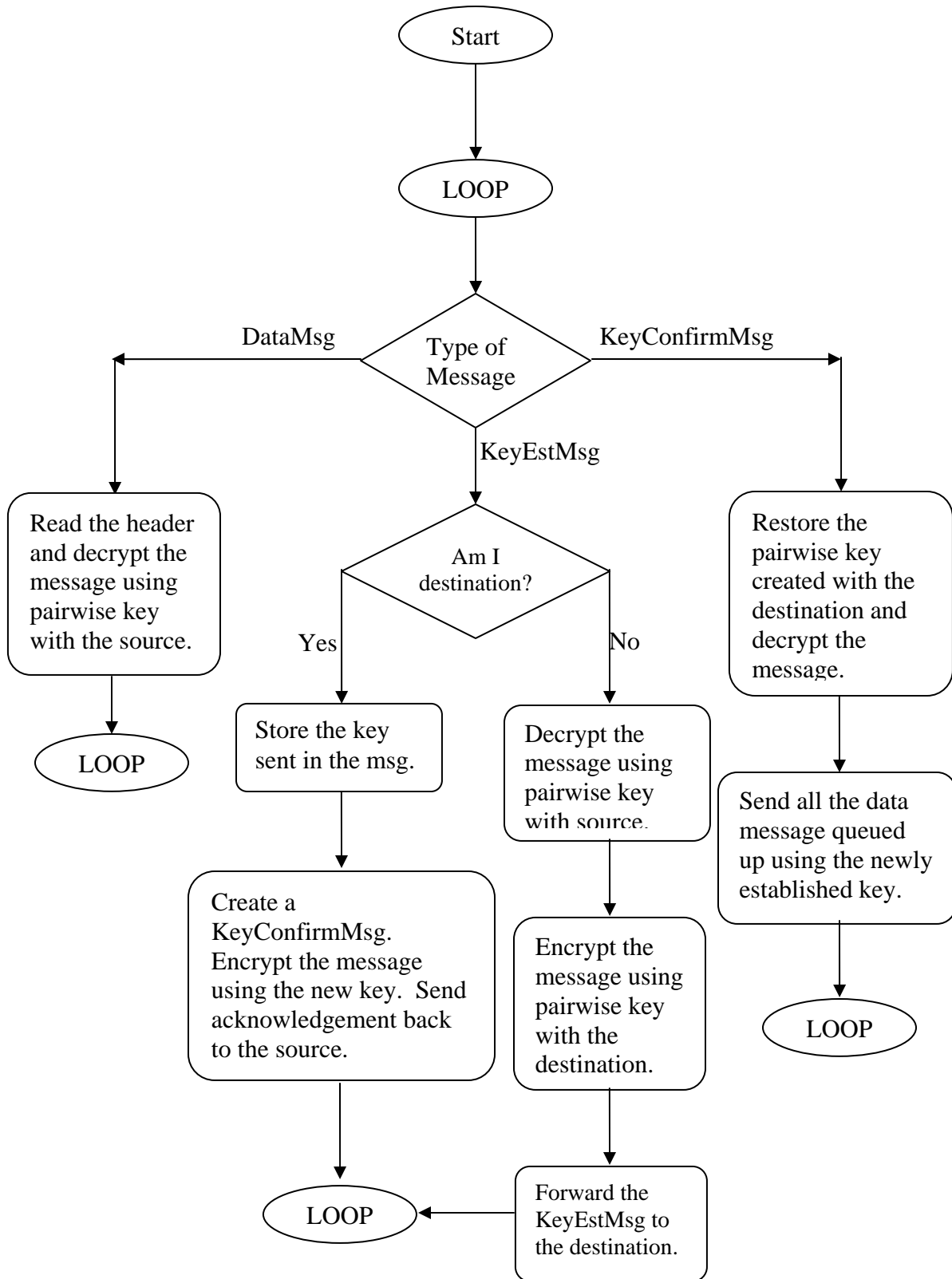


Figure 10: Flowchart for receiving thread

## 5. Possible Improvements

This implementation of PIKE needs  $2(\sqrt{n} - 1)$  keys per node. But the paper suggests a smarter implementation in which the memory overhead is halved. This scheme uses a pre-image resistant hash function. It suggests that for each node  $b = (a + i) \bmod \sqrt{n}$ ,  $i$  belongs to  $\{1, \dots, (\sqrt{n} - 1)/2\}$  assign a key  $h(K_a \parallel h(b))$  for pair  $(a, b)$  where  $K_a$  is a unique secret key for each node  $a$  from  $\{0, \dots, \sqrt{n}\}$ . Then for each other node  $c = (a - i) \bmod \sqrt{n}$  where  $i$  belongs to  $\{1, \dots, (\sqrt{n}-1)/2\}$  symmetrically assign the value  $h(K_c \parallel h(a))$  as the key for the pair  $(a, c)$ . Only pairwise keys with the other  $(\sqrt{n}-1)/2$  nodes need to be stored, thus reducing the memory overhead by factor of two. The following figure illustrates the keys generated for 25 nodes.

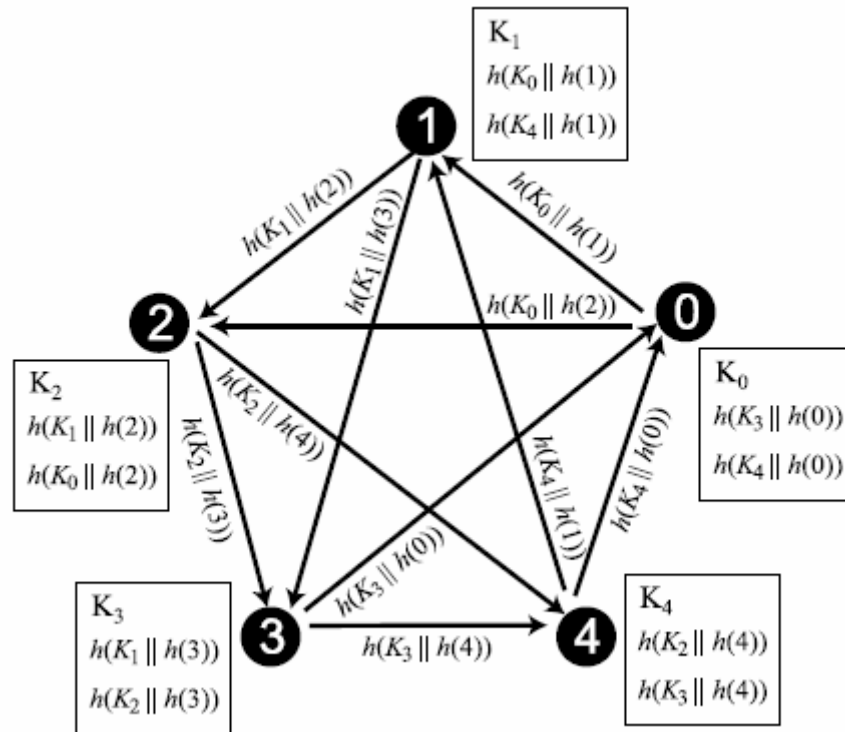


Figure 11: Pairwise key derivation for 5 nodes. Nodes are indicated as black circles. Each edge represents a unique pairwise key between two nodes, and the derivation of the key is indicated alongside the edge. Boxes next to the nodes denote the keys that must be stored on that node. For  $2k + 1$  nodes, only  $k + 1$  keys need to be stored.

This implementation of PIKE is done at the application layer in the sun SPOTs. One possible improvement could be implementing this protocol just above the routing layer.

This would mean that the top layer application can be independent of key establishment protocol. The packets being sent across would need to be modified to match the requirements of key establishment protocol and routing algorithm requirements. Such an implementation at lower layer can be a lot more extensible since the applications using key establishment protocols need not be changed.

## 6. Conclusion

One of the goals for developing PIKE was to achieve sub-linear trend in communications overhead as the network size increases since random key predistribution schemes are unable to achieve this. The simulations results provided in the paper show that PIKE is successfully able to achieve its goal while keeping the property of network resilience. The following graph shows that communication overhead varying with network density.

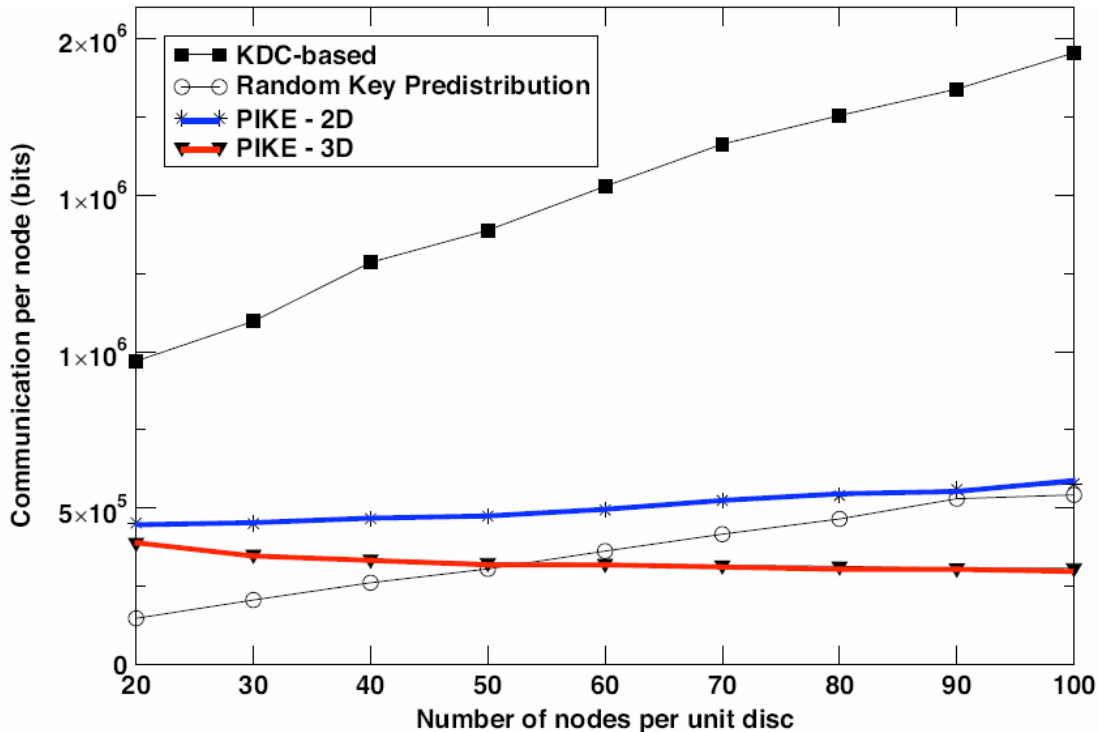


Figure 12: Communication vs. network density, network size = 5000 nodes

As the size of the network increases the amount of key establishment traffic also increases. In the KDC based scheme this increase is directly visible as the communication overhead steadily rises. For the random key distribution schemes the increasing the network sizes increase the number of broadcast messages and hence it surpasses the overhead of PIKE-2D at high densities. The communications overhead of PIKE scheme remains almost constant for varying node densities.

PIKE scales sub-linearly in terms of communication as well as memory overhead and maintains node resilience. It is not probabilistic and guarantees connectivity. The implementation of PIKE was done on top of AODV and runs successfully on Sun SPOT devices.

## 7. References

[1]. “PIKE: Peer Intermediaries for Key Establishment in Sensor Networks” by Haowen Chan and Adrian Perrig.

Paper appears in: INFOCOM 2005, 24<sup>th</sup> Annual Joint Conference of IEEE Computer and Communications Societies, Proceedings IEEE.

[2] “A pairwise key pre-distribution scheme for wireless sensor networks”, by W. Du, J. Deng, Y. Han and P. Varshney.

Paper appears in: Proceedings of the Tenth ACM conference on Computer and Communications Security (CCS 2003), pages 42-51, October 2003.

[3] “Security in Wireless Sensor Networks” by Adrian Perrig, John Stankovic, and David Wagner.

Article appears in: Communications of the ACM, Volume 47, Issue 6 (June 2004).

[4] “Improved Pairwise Key Establishment for Wireless Sensor Networks”, by Yi Cheng and Dharma P. Agrawal.

Paper appears in: pp. 442-449 of 2006 IEEE International conference on Wireless and Mobile Computing, Networking and communications 2006.