1 Introduction

When deployed to a disaster-stricken area, first responders need to be able to efficiently communicate their location and other information to their base camp. However, communication infrastructure is often unreliable or inaccessible during these crises. To solve this problem, we design an ad-hoc wireless network of portable devices that allows responders to send messages to their base camp in a reliable, efficient, and safe way. Our rigorous design accounts for the rapid changes in network topology that occur as the responders move around, as well as the possible presence of malicious agents in the field.

1.1 Design Overview

1.2 Tradeoffs and Design Decisions

2 Design

2.1 Routing tables

Initially, and every thirty seconds onwards, the network undergoes the following update procedure that discovers changes in the network topology:

- Each node issues one scan() message to populate its neighbor table.
- The base sends a route update broadcast() message to its neighbors, notifying them they are one hop away from the base. Each of these nodes updates their routing tables.
- Each of the neighbors from step 2 repeats that step themselves, aggregating the cost metric appropriately. Each node discards every broadcast() it receives after the first. This procedure continues until all nodes have received a broadcast, and takes time proportional to the diameter of the network.

The addition of new nodes to the network is also handled by this procedure. After this procedure, each node has multiple paths to the base and can begin using the metrics to estimate which paths are best.

2.2 Cost algorithm

2.3 Authentication

When designing a communication system for first responders, its important that first responders are able to trust the messages they are receiving from fellow first responders. Therefore it is important to establish a mechanism by which first responders can authenticate benign messages (i.e. messages not sent a malicious adversary). In the protocol presented in this paper, the base will use the RSA public-key cryptosystem to generate private and public-keys that it will then distribute to the first responders. In addition, everyone will have a table with an entry for each fellow first responder containing two important numbers:

1 The first responder's public-key.

2 The first responder's most recent message ID.

From here on, I will use the term node and first responders interchangeably.

2.3.1 The Table for Authentication

Recall that everyone will have a table with an entry for each a fellow node. The table will containing two important numbers for each node; the node's public-key and the node's latest message ID number. The latest message ID number is a counter-like number inserted to every message a node sends such that each node can idenfity ruplicate messages and be protected from malicious reply attacks. Consider Table 1 for a depiction of this table:

Table 1: Athentication Table

First Responder Number	Public-Key	Latest Message ID
0 [base]	69	669
1	50	837
2	47	877
3	45	300

2.3.2 Signing Mechanism

At the initialization phase of the Ad-hoc network, all nodes are required that they first report to the base before departing. At this step of the protocol, each of the nodes will be assigned their own private-key.
Furthermore, because at this step of the protocol the total size of the network is known, every node will be able to have an initial table with every public-key assigned so far. This will be quintessential for nodes' capability of verifying messages.
The system should accept messages only from first responders. Therefore it will be very important that nodes sign their messages with their private-key. Consider the following function that signs messages:

$$\text{SIGN}(message, my\_secret\_key) = \sigma_A$$

The function returns a binary string σ_A, corresponding to the output of signing the message with the node's secret key. Since each private key assigned for each node is unique, each other node will need the corresponding public key to verify the signature.

2.4 Authentication Mechanism

Even though it is important to sign messages for fellow nodes, its equally important that nodes are also able to verify the messages they receive. Therefore, consider the following verification function that takes a specific public-key and verifies if the chosen public-key matches the private-key used to sign the message:

$$\text{VERIFY}(signature, corooresponding\_public\_key) = b$$

Notice that when we say, matches, we mean that verify is indeed the inverse function of sign when the correct public-key is applied relative to the correct private-key used to sign the message. The output of VERIFY will be either, the boolean true if the verification passes with that public-key, or false if it doesn't. If the verification fails, then the message will not be forwarded.

The way that a node decides which public-keys to apply is simple, since it knows it neighboring nodes, it indexes the Authentication table using the node's number and then obtains the corresponding public key. Once the authentication has passed, then the node will use the following function to get the message:

$$\text{GET\_MESSAGE}(message, corresponding\_public\_key) = m$$

2.4.1 Dealing with reply attacks

_____

2.4.2 Scalability and updating the Authentication table

In summary, nodes will always apply their private-keys to sign any message they need to send. To verify and get the message, nodes will use the corresponding public-key to get the message and pass the verification.

3 Analysis

4 Conclusion

5 References