Securing the Rover System

Sulabh Agarwal, Kevin Kamel
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
sulabh@cs.umd.edu, kamelkev@umiacs.umd.edu

1 Introduction

Rover [1] is a system that provides location-based services to hand-held devices in a wireless environment. It is being developed in the MIND Lab at the University of Maryland. In addition to being location-aware, Rover is also device-aware, context-aware, user-aware and time-aware. A detailed description of Rover is presented in Section 2. The purpose of this project is to identify the security vulnerabilities in Rover and to design and implement solutions to mitigate them.

The basic design of the Rover System does not take into consideration any security issues. Security in Rover becomes complicated due to the following factors:

- *Wireless Medium:* A wireless layer is inherently broadcast oriented.
- *Low Power Devices:* The devices have limited power, so complex security encryptions cannot be executed on them [2].
- *Real-Time traffic:* The system can support real-time traffic, so the solutions have to be fast and efficient.
- *Many Users:* At any moment, the number of users in the system can be large, hence the security solution should be scalable.
- *Multi-Servers:* The design should be extensible to support multiple servers.

Rover is designed to function in a controlled environment. More specifically, there is a central control (Rover server) that all the clients communicate to. Moreover, all the users and devices are pre-registered with the server. These Rover design choices can be used as assumptions to develop a security module for Rover.

Since the Rover system can be widely deployed in museums, hospitals, schools, airports, disaster relief, etc, the security of Rover becomes extremely important. The rest of the paper is organized as follows. Section 2 provides a detailed overview of Rover. Section 3 presents an explanation of the protocols used in Rover. Section 4 analyses the security vulnerabilities of Rover. In Section 5, our solution is presented. Section 6 highlights the implementation and Section 7 concludes the paper.

2 Rover Overview

Rover is a system designed to provide location-based services to users in a wireless environment. It comprises of hand-held devices with an infrastructure to determine their location and provide them with services on the basis of their location. A possible scenario is groups of users with hand-held devices entering a museum. Each user logs into the Rover system and fetches a map of the museum. The map shows the user's location and updates it as the user moves (see Figure 1). When the user is near a display, the information related to that particular display pops up on the screen. The users can also establish voice chats with other visitors in the museum.

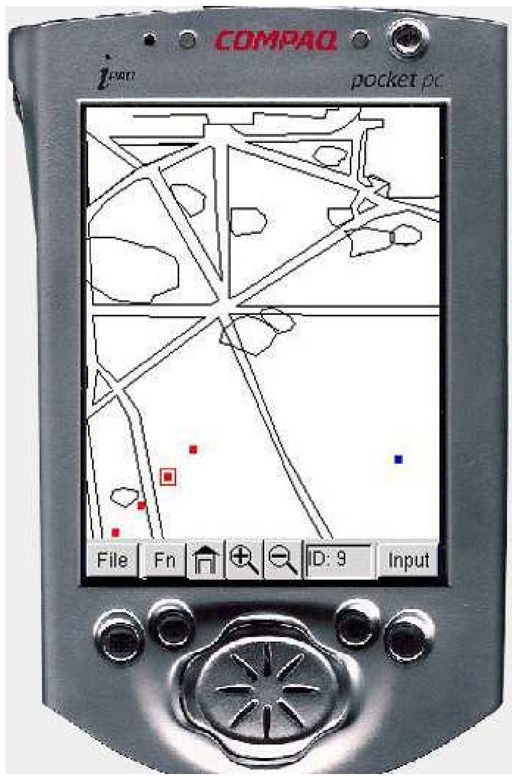

Figure 1: Map on a Client

The services that Rover provides to its users are as follows.

- *Basic data services*
  Services like text, multimedia, maps, directions, triggers, etc comprise the basic data services. Usually, this data flow is from the server to the user. The server has to modify the data to suit the device's capability. The user can customize the received information by selecting the desired options at his interface.

- *Transactional services*
  These services require co-ordination between the user and the server. E-commerce transaction is an example of transaction services.

A Rover system consists of various entities:

- *Users:* The people who use the system (eg. visitors in the museum)

- *Clients:* The clients are hand-held devices with wireless capabilities. They have limited processing, storage and power.

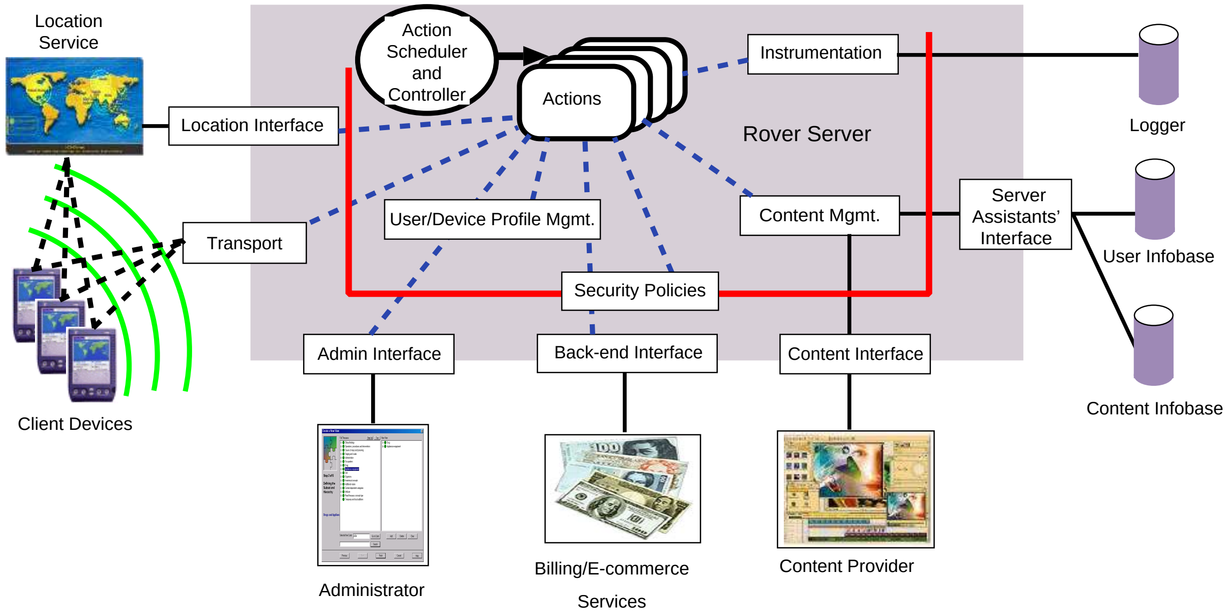

Figure 2: Rover architecture

- *Wireless infrastructure:* The connectivity among the clients and the server are maintained through a wireless infrastructure. The technology may be IEEE 802.11, bluetooth, etc.
- *Servers:* The servers are the central systems that manage the services provided to the users. Servers consist of the main controller, Location-server, Database, Logger, etc. Among these the clients communicate directly only with the main controller.

Figure 2 shows the layout of the Rover architecture.

2.1 Communication Protocol

Every user is registered with the server. Each user is given a user-name and password. When the client program is executed a TCP connection is established between the client and the server. This is a "control" connection. One such connection is maintained for every client in the system. All the communication for fetching information from the various server components is done through this channel. The client sends sensitive information like its user-name and password, and the location information over this channel. Any attacker eavesdropping on this channel can obtain access to the system by gaining knowledge of the password, can modify the location information of the client, etc. This is the channel that is also used for e-commerce transactions.

The users have the capability to establish multimedia (audio, video, etc) communication amongst themselves. Let us run through an example to understand how the communication is set up. Suppose user Bob wants to have a voice chat with user Alice. Bob sends a *start chat* request to the server over the tcp control channel. Bob also starts receiving on a udp port. The server checks if Alice is already chatting with somebody else (at the moment we do not have the support for multiple chats from the same user). If Alice is busy, the server sends a *reject chat* message to Bob, otherwise the server forwards the request to Alice. Alice can either accept or reject the chat and an appropriate message is sent to the server. If Alice accepts the chat it starts to receive and send voice packets over a udp port. The server then forwards this message back to Bob and marks Bob and Alice as talking to each other, if Alice accepted the chat. Bob on receiving the *accept chat* message starts sending to Alice.

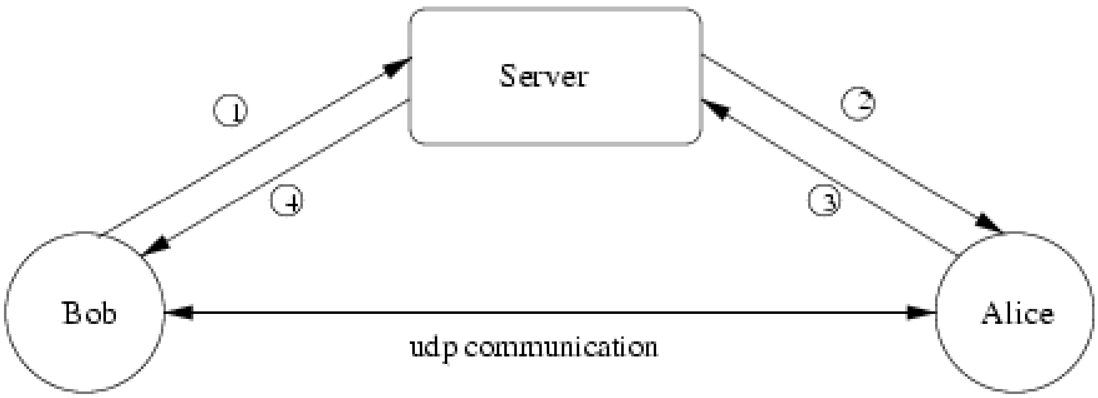

Figure 3: Chat control handshake

Figure 3 shows the complete handshaking to establish a chat between Bob and Alice. The detailed messages are:

- Message 1: $StartChat, Port_{Bob}, ID_{Alice}$
- Message 2: $StartChat, IP_{Bob}, Port_{Bob}, ID_{Bob}$
- Message 3: $AcceptChat, Port_{Alice}$
- Message 4: $AcceptChat, IP_{Alice}, Port_{Alice}$

After all this control communication through the server Bob and Alice send voice packets to each other over an insecure udp channel.

3 Rover vulnerabilities

Let us look at the security pitfalls in the context of Rover being used in a hospital. Every doctor has a hand-held device. All the patients that require attention are highlighted on the map and the doctor can visit each one of them. Whenever the doctor approaches a patient, the display provides information about the medical history of that patient. Also, the doctor can receive emergency messages, ask for medicines, etc from the client. Clearly, we would need this scenario to be secure so that any accident, violation of patient privacy, etc can be prevented.

The various threats that Rover is vulnerable to are as follows:

- **Denial of Service**
  There are two kinds of denial of services - at the physical layer and at the service layer. As a part of this project we will not deal with frequency jamming that can cause the physical layer to break down. In the event of a denial of service at the higher layers, the doctors will not be able to identify their patients, get information on them and will not be able to respond to emergency situations.

- **Unauthorized Access**
  A malicious user can sniff the password of another user and login as that user. The malicious user can then change the patient data, order improper medicine, etc.

- **No Confidentiality**
  A malicious user can sniff data and voice traffic over the network. He can possibly gain access to a patient's medical history and blackmail him.

- **No Integrity**
  An attacker can masquerade as an administrator and remove all the clients, devices, etc. Also, an attacker can change the data in a packet and such an attack will be difficult to detect.

As is outlined above, there are a number of threats that Rover has to guard against. Also, these attacks are easy and cheap to execute. Rover has no security built in so a simple packet sniffer will be able to accomplish most of the attacks. In the next section we present a solution to provide confidentiality, integrity and availability in Rover.

4 Solution

As was outlined in section 2, there are two insecure channels of communication in the Rover system - between a client and the server and between two clients. The connection between the client and the server is a TCP connection and it provides constant connectivity. This is a highly important channel as it is used to transfer confidential information (passwords, credit card transactions, etc). We would like to add relatively strong security to this channel. At the same time, the clients are low powered devices and hence we would prefer a light-weight solution. A credible solution works as follows.

The server is assigned a private key and a public key certificate for that private key. Instead of establishing an insecure TCP communication with the client, we run Secure Sockets Layer (SSL) over the TCP. This builds a security layer. SSL is a session based protocol and hence works well with a connection-oriented transport layer. Thus, SSL clearly fits the model that we are working with. The client and the server decide upon the session key and the ciphers to use through an initial handshaking. The session key decision is secured by using the server's key pair. Once the connection is established it is secured by the ciphers and checksums that the client and the server decide.

A problem related to this approach is that there is no initial authentication of the client. In this aspect, the handshaking here is similar to the one done when a browser accesses a server over the Internet through a secure channel. However, since each user is pre-registered with the Rover system, every user has a password that can be used to authenticate the user once a secure channel has been set up. The cipher that the SSL connection uses defaults to DES_CBC3_MD5. More details on SSL handshaking and data transfer over SSL are presented in Appendix A.

The peer to peer communication uses an insecure UDP communication. SSL is not used to secure this channel for a variety of reasons. SSL is a session based protocol and hence inherently unsuitable for use over UDP. The chats are real-time communication and hence we desire an extremely light-weight solution. In the scenarios that we envision, the multimedia data will always require a lesser degree of confidentiality than the client-server communication. Providing data and source integrity requires more computation at each end and that would incur a high processing overhead for the clients.

Due to all of these factors we decided to use some simple symmetric cryptographic scheme to provide basic confidentiality of the voice data. DES encryption is used to encrypt the voice packets. The DES key is generated by the client that initiates the chat. This key is then transfered during the chat handshaking to the peer and both the users then have the same key. The session key interchange is secure since that happens over the SSL connections between the server and each of the clients. The handshaking is the same as shown in Figure 3. The detailed messages exchanged are:

- Message 1: $StartChat, Port_{Bob}, ID_{Alice}, key$
- Message 2: $StartChat, IP_{Bob}, Port_{Bob}, ID_{Bob}, key$
- Message 3: $AcceptChat, Port_{Alice}$
- Message 4: $AcceptChat, IP_{Alice}, Port_{Alice}$

After the handshaking each of the clients have the same key. The key is exposed to the server. However, the server is completely trusted in our model. Moreover, it is a machine with higher capabilities and hence is more secure. The clients encrypt the voice packets using DES in ECB mode and the receiving client decrypts the packets in the same way. Although there are other more secure modes of DES available, they are too heavy for our real-time communication. DES in CBC mode was implemented and experimented with for a voice chat and there was a considerable degradation of the sound quality. We believe that ECB mode strikes the right balance between security and performance. Appendix B explains the DES algorithm in detail.

The designs presented above overcome the initial complications that we outlined in Section 1. Our solution is light-weight, efficient, scalable and easy to implement. We have tried to strike a balance between security and overhead in terms of computation and network usage.

5 Implementation

The current implementation of Rover has been developed on the Linux OS. The server is running on RedHat7.1 on x86 machines. The clients are Compaq IPAQ Pocket PC (H3650) running the familiar linux distribution v.0.5. The wireless technology being used is IEEE 802.11 WLAN. Each IPAQ has a PCMCIA 802.11 card attached to it.

The clients are cross compiled for the arm-processor of the IPAQs using the skiff toolchains. The clients link with libssl.0.9.6 and libcrypto.0.9.6 that are cross compiled for the IPAQs and available on [5].

The server uses an RSA key and a certificate. For secure multimedia communication the client generates a 64-bit random key. The client seeds a PRNG with the time of the day and generates a random value. This value is then used to compute a character. The client generates an entire string in this manner. This string is then used to seed the PRNG of the openssl library, after which a random key is generated by calling `des_random_key` . This function returns a 64-bit DES key.

6 Conclusion

Rover is an infrastructure that has the potential to be used in various critical and conventional settings. It is imperative for Rover to have security built into it. As a goal of this project we have presented a design to add security to the existing infrastructure. A discussion has been provided to justify the design decisions and a subsequent implementation has been performed to confirm the theoretical arguments. We believe that this work provides considerable security to Rover.

References

[1] Suman Banerjee, A. Agrawala, A.U. Shankar, *et al*, "Rover Technology: Enabling a Location-Aware Computing Environment", *CS-TR 4312, Dept. of Computer Science, University of Maryland, College Park*, July 2001.

[2] Armando Fox, Steven Gribble, "Security on the Move: Indirect Authentication Using Kerberos", *Proceedings Second ACM Conf. on Mobile Computing (MobiCom96)*, White Plains NY, USA. 1996.

[3] M. Bishop, "Computer Security", TextBook.

[4] http://www.openssl.org

[5] http://www.handhelds.org

[6] http://www.itl.nist.gov/fipspubs/fip46-2.htm

A Secure Sockets Layer

Secure Sockets Layer provides security over connections and sessions between client and server. A session is an association between peers and a connection is the set of mechanisms used to transport data in a session. A session can have many connections. The data associated with a session includes:

- a session identifier
- a compression method to reduce the data
- a cipher specification for the encryption and MACs
- a master secret of 48 bits

All of the above information is stored both at the client and the server involved in the communication. Besides this there is some information that is connection specific:

- random data for the client and the server
- server and client data encryption keys
- server and client MAC keys
- IVs for the ciphers

- server and client sequence numbers

SSL consist of two phases. The initial phase uses public-key cryptography to exchange session keys through the handshake protocol. The messages are enciphered using a classical cipher and checksummed using a cryptographic checksum. The Lower Layer of SSL, known as the record layer, provides both confidentiality and integrity of the messages being sent. It runs on top of TCP. The record layer uses the cryptosystem negotiated by the handshaking. Each message is compressed, MAC is computed and the message and the MAC are encrypted. An SSL record header is attached to the resulting block.

The SSL handshake protocol sets up the parameters needed for the SSL record protocol. It consists of four rounds to agree upon keys, ciphers, and MAC algorithms. The first round creates the SSL connection and the cipher and the compression mechanisms are decided. In the second round the server authenticates itself. In round 3 the client validates the server's certificate and sends its certificate to the server, if requested. The client and the server also compute the master secret at the end of this round. The fourth round consists of acknowledgements between the server and the client. After this the encrypted communication can begin.

B DES Encryption

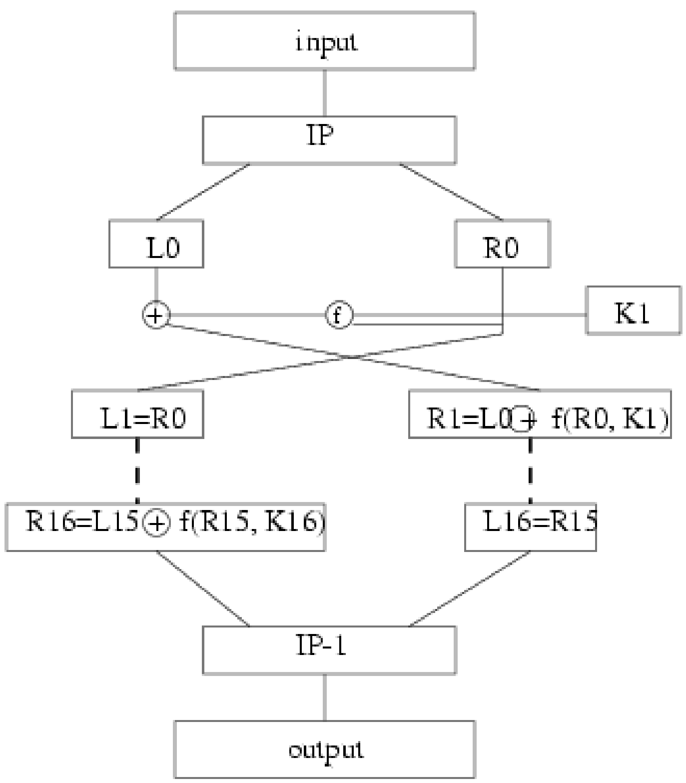

Figure 4: DES Encryption

In the Data Encryption Standard (DES) the key, input and output are all 64 bits long. The cipher consists of 16 rounds. There is a separate key of 48 bits for every round. The 48 bit key is generated by dropping the parity bits from the key and then permuting the bits and extracting 48 bits from it. Different bits are extracted in each round, thus resulting in different keys for each round.

The output of a round is the input of the next round. The rounds are executed one after the other. Figure 4 shows the DES encryption. The right most 32 bits of the input and the round key are passed through a function f and the result is xor'ed with the left 32 bits of the input. The left and the right halves are then swapped.

The function f takes the 32 bit input and expands it to 48 bits. These 48 bits are xor'ed with the input key and the result is split into eight 6-bit sets. Each of these sets is put through a substitution table and each of them produces 4 bits of output. These are then joined to produce 32 bits that is then permuted to generate a 32 bit output.

When DES is used directly it is called as being used in the Electronic Code Book mode (ECB). The other mode that DES is being used in this project is Cipher Block Chaining mode (CBC). In CBC each block of ciphertext also depends on the previous ciphertext block.

DES is not a very strong cipher. Due to this reason it is used to encipher sensitive but non-critical information. Thus, it is suitable to encrypt the multimedia communication in Rover using DES.