Rethinking Mobile Telephony with the IMP

M. DeYoung, N. Henke, G.Wai and J. M. Smith
University of Pennsylvania

Abstract **: The recent widespread deployment of wireless LAN technology raises the question of how a mobile telephony system might instead be architected to use wireless LAN access points and the Internet to achieve similar services. In this paper, we examine an end-to-end architecture for mobile telephony, with a strong focus on endpoint issues.**

We have designed, implemented, and have experience using devices we call Internet Mobile Phones or IMPs. The IMP system provides encrypted wireless voice communication over 802.11B LANs. IMPs run Linux on a lightweight single-board computer running customized voice over IP software; data is encrypted with 128-bit Blowfish.

The paper reports on our design decisions and the resulting implementation of the IMPs, with sufficient detail to reproduce the devices. We report our experiences with using them for several months in a laboratory environment, and close with proposals for future experiments to investigate scale and extensibility.

1. Introduction

The increasing ubiquity of Internet communication and wireless access points has made universal Internet access increasingly convenient. There are clearly barriers to this vision of accessing a device by its address such as NAT boxes, firewalls and for-pay services, but social factors and technologies such as IPv6 favor its long-term growth. A potentially important consequence of this infrastructure is the ability to utilize the Internet for new applications, some of which are enabled simply by the availability of new edge devices [6] and the addition of some software "glue" to achieve interoperability with the existing Internet infrastructure.

The cellular telephone system provides widespread wireless voice service, and is integrated with the traditional telephony system. This enjoys excellent "network externalities", as the first deployed cellular telephone is part of a network with extensive connectivity. Cellular telephones are now evolving towards computers, but the basic bearer architecture has remained one of telephony. *Table I* illustrates the design choices for infrastructures for mobile telephony, with access and bearer architectures on one axis and some common devices on the other. The traditional mobile phone system uses circuit-switched access and a circuit-switched bearer architecture. While provision of VOIP bearer services is underway in the telephony industry, and portions of it such as SS7 or SCTP signaling are packet-switched, deployed mobile telephony access infrastructure is circuit-switched. 3GPP and 3GPP2 phones resemble Internet terminals, but use a managed guaranteed bandwidth network.

Device Type	*Mobile Access Infrastructure*	*Bearer Service Infrastructure*
Cellular Telephone	Wireless Circuit-Switched	Circuit-Switched (VOIP core?)
Internet Mobile Phone (IMP)	Wireless Packet-Switched IP	IP packet switched

Table *I: Switching and Roles in* Voice *Services*

One timely question is whether the growing Internet infrastructure is well-suited to providing a mobile telephony service, now that wireless access through 802.11b LANS is increasingly common, and often publicly accessible.

While a widespread IP telephony infrastructure is now being developed and deployed with various Voice Over IP (VOIP) protocols, reproducing various features of handheld wireless voice devices intended to behave as mobile telephones presented an interesting design exercise. In addition, since features are added at the endpoints in the IP architecture, design choices existed for privacy protections unavailable to the traditional cellular telephone user. These protections are particularly attractive in the context of wireless VOIP services.

1.1. Experimental Goals

Our hypothesis was that a handheld mobile 802.11 device using IP as a bearer service would provide indications that an alternative architecture for mobile telephony was possible if not today, then in the very near future. Our experimental methodology was the implementation of a system, largely centered around the design of the endpoint (as the end-to-end argument [21] would suggest is the appropriate design objective) we call the Internet Mobile Phone, or IMP. An operational IMP which emulates circuit-based telephony over wireless access to IP presents an objective demonstration of the strength of a purely-IP based mobile telephony approach.

Our high level design goals for the IMP were as follows:

1. Design and implement a user device which would serve as a portable personal 802.11b client.
2. Implement a fully capable IP host on the device, that given an IP address would be able to interact with any other, including another such device
3. Provide a simple Voice Over IP (VOIP) service

4. Provide as much of the ad-hoc, unregulated "feel" of a walkie-talkie as possible, while providing a reasonable mobile telephony service
5. Provide a secure channel establishment protocol and encryption of the data on the VOIP channel for privacy protection.

1.2. Outline of this paper

This paper is *not* about the number of such devices that can be supported by an 802.11b access point, nor is it about the quality of service required [1] or voice quality achievable. It is not about audio for desktop use on a LAN [23,24,25]. Instead, we sought to prototype a working secure IP mobile telephony system that met our high level goals, and was realizable at a reasonable cost in both time and money. We believe we have achieved that, as we used the devices to communicate at what, by virtue of our many conversations we judge to be reasonable voice quality, at least as good as the commercial cellular telephones we use. As an aside, since some of the conversations were about the system, it might be argued that it helped bootstrap itself into existence.

The next section, **Section 2**, provides a more detailed discussion of the design goals for an IMP. **Section 3** describes the hardware realization of our handset, including a parts list for those interested in reproducing our experiment (one of our long-term research goals is to build a user community to test scalability issues – see **Section 8**). **Section 4** discusses the software environment, which required substantial engineering of device support software and integration of encryption and VOIP. **Section 5** describes the implementation and evaluation of the security features of our system. **Section 6** describes our experience using the system. **Section 7** concludes the paper with our assessments of the contributions of this work and desirable directions for follow-on work.

2. Detailed Design Goals

An Internet Mobile Phone is a mobile voice communication device that uses the increasingly prevalent wireless LAN access to IP. An IMP experimental prototype must address the following design issues:

- IP based wireless networking
- Security Architecture and Authentication
- Voice Communication System
- Device Operating Environment
- Hardware and packaging

IP based networking is a central design goal as it allows us to address the larger hypothesis of the paper, that wireless LAN technology provides a natural transition from circuit-centric to IP-centric mobile telephony. The 802.11b wireless LAN is widely deployed, and IP addresses are obtained at wireless LAN access points that interface to the broader Internet, typically via DHCP. IP transport can be used for both the control and data planes of the system.

Security criteria can be modeled as *CIA – Confidentiality, Integrity* and *Availability*. Any broadcast network can be disrupted by transmission-based attacks (such as jamming or flooding) which would inhibit availability. Thus we did not make availability a security goal, but did set confidentiality and integrity as major goals. The use of cryptography seemed obvious; the question is at where in the design it would be employed. In brief, we chose an end-to-end strategy and encrypted voice before transporting it via conventional IP. Some privacy goals (such as knowledge of who is talking to whom) are sacrificed in the service of sensible layering, although use of an IP-based service allows one to employ very sophisticated schemes (such as those based on IPsec tunnels [28] or onion routing) if so required.

The IMP establishes logically peer-to-peer connections to transmit voice data. Since dynamic IP addressing is used, and IP communication between peers requires at a minimum reciprocal knowledge of addresses, we maintain a list of the IP addresses of all connected users. When a user logs on to the system the authentication system will store the IP address assigned to the user and allow the user to retrieve IP address of other connected users. This has the flavor of a "buddy list" or simply an on-line directory. A variety of other schemes are possible under various assumptions about how addresses are managed, such as per-device static IP addresses; beaconing with ring search for IP address discovery using fast broadcast [7]; or the "home address" concepts used in Mobile IP [11]. These issues are not central to the architectural question and while technically fascinating are beyond the scope of this paper.

There are a variety of Voice Over IP (VOIP) platforms. One such platform, the Nautilus [4] open source system includes encryption, provides a good basis for securing a peer-to-peer voice channel, using Diffie-Hellman [3] key exchange for each conversational session. We note in passing that Diffie-Hellman key exchange requires no external method of key management.

Easy implementation of the prototype suggested a general purpose computing platform with a familiar programming environment, especially as much of the work required in integrating the components of the system would be done in software. The wide range of devices supported by Linux, the availability of the Nautilus system, and a variety of embedded versions appropriate for single-board computer (SBC) implementation motivated our choice. Among other tasks to be detailed in **Section 4**, the software manages the bootstrap, authentication/contact lists, user interface, call setup and teardown, networking, the voice codec and encryption.

Our hardware design goals were first to use as many off-the-shelf components as possible, and to compose them to both ease prototyping and to meet the packaging constraints of a system intended for handheld mobile use. While a secondary goal, we also

sought to be reasonably cost-effective in our choices. This suggested, for example, that we choose a design which could accommodate 802.11 hardware readily found in our laboratory, such as PC-MCIA implementations, and this influenced our choice of SBC. As discussed in the next section, an IMP can be realized for under $1000 by a capable hobbyist.

3. Hardware Design and Realization

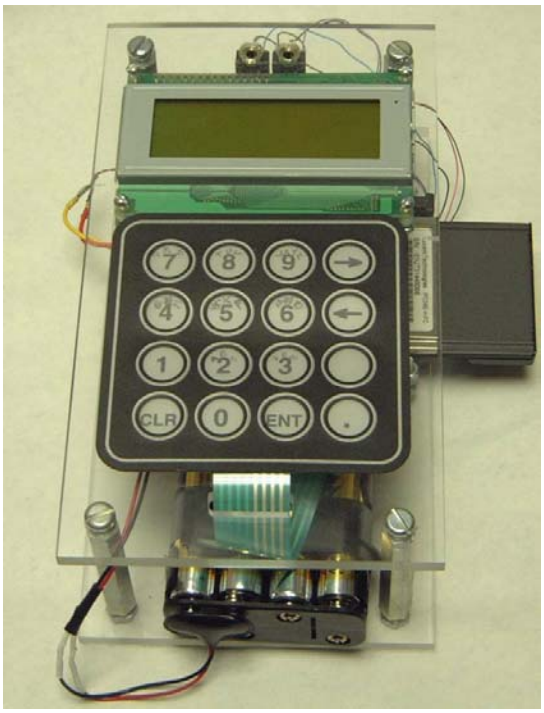

Figure 1: Top View of the IMP

The hardware is small enough for handheld operation. We used a small, battery-powered single board computer (SBC) based on the StrongArm used in many mobile telephones, the Elf [10]. We operated the Elf at 133 Mhz, although the system's data sheet indicated operation up to 206 Mhz. The processor is

Table 2 : Components used and costs

Item	Manufacturer	Cost
Elf SBC	Inhand Electronics	$640
Keypad	Digikey	$15
LCD display	Matrix Orbital	$85
802.11bPCMCIA	Orinoco	$90
Plexiglas Case	Machine Shop	$120
8 AA Ni-MH Battery	Energizer	$24

capable of 150 Dhrystone MIPS at the 133 Mhz clock rate. The Elf requires battery power in the range of 6.0 volts (drawing 1.5 Amps) to 12.0 V (drawing 1.0 Amp). A pack of eight AA Nickel-Metal Hydride rechargeable batteries was used. These batteries provide 1.2 V each, for a total of 9.6 V. Each battery has 1700 milliamp hours (mAh) of power, and though the draw varies with use, experience showed that this eight-battery configuration normally lasts over two and a half hours while the phone is being used continuously, and a longer lifetime if the IMP is idle. We have made no serious attempts in our prototype to address power management, but as with packaging, the problem can be addressed ; see e.g., [8]. An LCD display [14] is used for output, a keypad for input. The LCD display and keypad communicate with the Elf through an RS-232 serial cable.

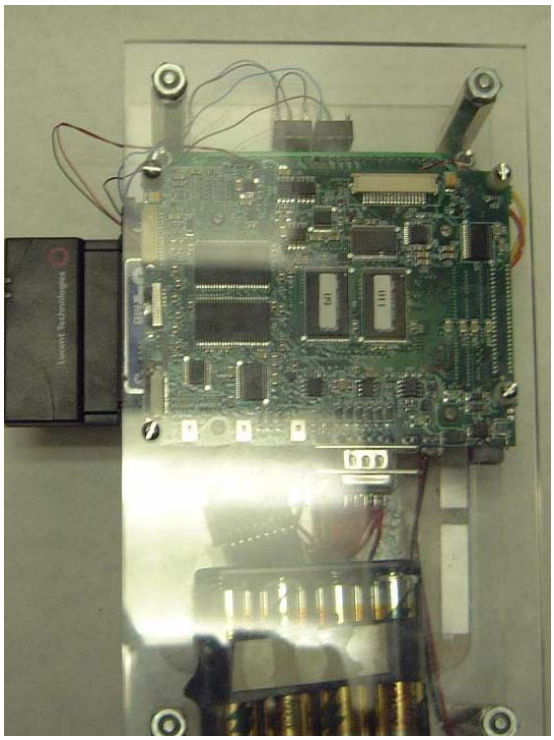

Figure 2 : Bottom View of the IMP

The SBC is coupled to commercial PC-MCIA 802.11b devices. The channel bandwidth must be great enough to transmit voice data. The most compact encoding scheme that may be se dis LPC-10, which requires 14.4 kbps plus some overhead. The data rate must be greater than approximately 15kbps. IEEE 802.11b supports much higher data rates (11 Mbps) so this is not an issue. Throughput tests using ttcp between the Elf and a more powerful computer indicate that the IP/802.11 system can obtain more than adequate throughputs of 500 kilobytes per second. The use of commercial off-the-shelf devices means that any commercial 802.11b access point can be used as infrastructure for the device. Parts are detailed in *Table 2*.

Using these components, the total cost of implementation is well under $1000, in reach of many hobbyists (we purchased one of the Elfs as part of a development kit costing about $3000, but the additional $2400 for the development software we consider a one-time cost, as others can use our GPLed

source). As the device was intended for mobile use, the individual parts were packaged into a portable unit, as photographed in *Figures 1* and *2*.

Two Plexiglas panels house the components. The bottom panel (*Figure 2*) holds the SBC and the battery pack (which can be seen at the bottom of the photograph), while the top panel (as can be seen in *Figure 1*) holds the LCD, keypad, and microphone/speaker jacks. The overall size is 8.5" long x 4" wide x 3.5" high – the size can be inferred from the familiar AA batteries or 802.11 card. The Plexiglas frame was milled by a local machine shop.

The SBC is located on the bottom Plexiglas panel, as can be seen in *Figure 2*. A specialized implementation of the Linux operating system runs on the SBC, which manages the user interface and secure voice over IP software. The software architecture is described below in **Section 4**. To ease implementation the database is run remotely on a server, as we describe in **Section 5**.

The system has proven to be quite reliable, in spite of stress-testing from laboratory clumsiness and transport in backpacks and coat pockets.

4. Software Design and Realization

Software systems comprised the majority of the effort in prototyping the Internet Mobile Phone. The effort was split up into the encryption software, the voice over IP software, the authentication software, and the phone operating software. The encryption and voice over IP software were based on the Nautilus open source software. In addition, the software also supports the following features:

- Full Duplex Audio (for good interactive use)
- User Modifiable Database
- Secure Voice over IP
- LCD and Keypad Interface
- Stateless Client Device

The basic flow of the device software is straightforward. It initializes the device (essentially a bootstrap) when the device is powered up, and then waits for a user login. Data is sent to the server to authenticate for local access. A contact list of IP addresses is retrieved and the system is then ready to send or receive voice calls. In either the caller or callee mode, Diffie-Hellman key negotiation is used to generate a session key used to encrypt the voice conversation. Call termination results in the system returning to the "prepared for call" state.

4.1. Operating System

Linux has been ported [13] to run on the Elf's StrongARM processor, and drivers have been written for much of the common hardware that populates the embedded boards that have these processors. As mentioned before, the Elf SBC is constructed on the Intel StrongArm processor platform. Our development environment, however, operates on Intel x86 processor platforms, so cross-compilation was required. We incrementally built a language development environment for the Elf and then moved other software as required.

We use the Boot Loader OBject (BLOB) boot loader to load and start the operating system. We had to modify BLOB to run on the Elf, mainly modifying addresses at which memory was located as well as sizes for this memory. This included SDRAM chips to initialize the Elf RAM, as well as onboard Flash in which the kernel and embedded filesystem are stored. To ease operation, we configured the serial hardware on the Elf so that a remote terminal could connect to BLOB, through which commands could be entered via a text console. These changes allow BLOB to locate and run the Linux operating system.

4.2. Device Support

The Elf has a Phillips UCB1200 audio codec [17] on its board. To use this hardware to record and play audio the Linux driver source code was modified, mainly to choose appropriate initialization parameters such as IRQ and I/O ports, but also to issue commands directly to the codec hardware to correctly initialize the hardware buffers.

The Elf also includes an embedded PCMCIA controller chip on the board. The StrongArm SA1100 processor chipset provides a method to connect each of the PCMCIA card sockets on the PCMCIA controller directly to separate locations in DRAM, and grant access to the controller chip directly by writing to standard memory locations. Unfortunately, the Elf does not use this method of interacting with its PCMCIA chip. Rather, it is manufactured with both of the PCMCIA card sockets mapped to the first DRAM address, instead of separating the sockets to different DRAM locations. In addition to this alteration, the Cirrus chipset is only supported by the Linux kernel when it resides on an ISA or PCI bus. Due to the attachment of the controller chip directly to DRAM, it was necessary to change the driver to directly access the controller chip registers through memory. The changes to facilitate this were provided for an earlier version of Linux, and they were fixed to work on the most current versions of the Linux kernel. We also had to virtualize addresses for memory mapped I/O. Once the PCMCIA cards themselves were accessible, the Linux PCMCIA driver table was configured to access the cards.

A modified Linux was needed to provide the software platform on which the SVOIP program as well as the user interface could run. The Elf has 16 MB of DRAM available, which after system startup leaves between 5 and 8 MB for user programs. The base of any version of Linux is the combination of libraries that are provided. The programs that were run on the Elf were queried for their library needs, and a minimal list of libraries compiled. These libraries were then stripped of all debugging support and extra data.

To run user commands and provide the basic system commands, Busybox was used. Busybox is an

embedded program that provides all of the functionality of the standard Linux system commands in one monolithic program. Busybox is configurable to include only minimal necessary functions and programs. The final packages that were added to Linux were the serial line file transfer package, lrzsz, and the standard Linux PCMCIA packages that provide programs to control the drivers and cards.

The base filesystem was populated with these libraries and programs, and it occupied 6 MB. Unfortunately, the maximum allowable size is 4 MB. As there are two available PCMCIA slots, the second slot was used for a 64MB flash card with an IDE hard drive interface. To reduce the space used, the programs that are needed to mount an IDE disk card were queried for their libraries, and all but those few libraries were moved to the IDE disk. Symbolic links were used to point from the standard library locations to the new locations on the IDE disk. These links are "dangling" until the IDE disk is mounted, but none of the programs that use those libraries are run until the disk is mounted, ensuring reachability when needed. This trick reduced the base filesystem size to 2.5 MB.

All code patches were submitted back to the package authors for inclusion into the main code base. In the case of the audio driver, it was accepted into the main Linux kernel. This allows anyone with an Elf to download the Linux kernel and use the audio hardware. Patches have also been made available to the Elf Linux support group. The drivers are in use by this group, and are also being used to start the official Linux support from the Elf's manufacturer.

4.3. Voice over IP Architecture

The intentionally simplistic voice over IP software subsystem is comprised of two parts:

1. Voice encoding/decoding
2. Transport layer protocol

As discussed in the previous subsection, the Elf SBC includes the Phillips UCB 1200 single chip integrated mixed signal audio and telecom codec. The UCB 1200 allows for programmable sampling rates, so the sampling is set by the voice encoding software. The voice encoding software controls audio sampling and encoding. The software records samples from the audio hardware and encodes the voice using an existing voice codec. The voice codec digitizes analog voice data from the microphone into a readily transportable form. The voice encoding software is also responsible for reversing these steps at the receiving device.

Once the voice is encoded into binary form and encrypted with the encryption software, the data must be packaged and sent to the receiving party. The voice over IP software uses two Internet data channels to conduct the conversation. TCP is used for a reliable control plane and UDP is used for voice packets under the assumption that humans can interpolate data losses.

The voice samples were 16bits, and would ideally be taken in 320 byte frames at 8Khz. As the codec limited sample frames to powers of 2, we used 512 byte frames.

While the base Nautilus system had many desirable features, the audio hardware control and Internet data transport mechanisms are not appropriate. The software does not allow full duplex audio, essential for voice conversation. The data transport protocol is proprietary, and built on UDP/IP. This appeared unnecessarily complex; we simplified it for engineering and performance reasons.

The software is able to handle either an unencrypted or fully encrypted, full duplex voice conversation. When a call is received, the SVOIP software is initialized. Parameters are exchanged to make sure both parties want the same level of encryption, to identify the port used, and to ensure compatibility of other session parameters. If it is an encrypted call, the Diffie-Hellman protocol is used to negotiate a secret key for the duration of the call. The software is then multithreaded using POSIX threads. This is called the audio loop.

The audio loop encodes the voice data, sends the data, receives data, and decodes it. This has been added to provide full duplex audio. Full duplex audio is achieved by spawning three threads. The first thread reads data from the microphone, encodes it, encrypts it, and sends it to the other device. The second thread waits to receive data, decrypts it, decodes it, and plays it through the speaker. The third thread is used for a control signal; once the call is ended, the program is exited and the user is returned to the user interface.

The voice data is encoded and decoded using the G.711 μlaw codec. This codec does not compress, and therefore provides the highest possible audio quality. The bit rate transmitted is about 25 kbps. Wireless 802.11b allows for up to 11 Mbps, and one antenna has a range of 150 to 200 feet. Many more users could share an access point, even in this simplistic mode, which has the desirable properties of good quality and low delay. Some crude measurements indicate that the 512 byte frames are encoded and encrypted in about 40-45 ms. There is no noticeable difference in delay with the encryption on or off, suggesting that the delays in the system are from buffer copying from the audio hardware into a user space buffer.

The main control signal sent besides initial parameter exchange is that to end a call. The third thread in the audio loop sits and waits for a "kill" signal. When one party ends a call, it joins the threads, sends a "kill" signal through the TCP channel, and then closes all open file descriptors and sockets. Similarly, once the other party receives this "kill" signal, the threads are joined and all open file descriptors and sockets are closed.

5. Security Architecture and Evaluation

As we discussed in **Section 2**, of the "CIA" criteria our design goals for the security architecture included confidentiality and integrity for voice conversations, but did not include availability. Our reasoning was that the spectrum management and multiplexing issues are addressable elsewhere in an IP-independent fashion,

such as via spread-spectrum for the former concerns and via overprovisioning or QoS for the latter.

5.1. Authentication

Most IEEE 802.11b infrastructures use dynamic IP addresses obtained via DHCP. A directory service must exist for users to determine each other's addresses for peer-to-peer connection. As mentioned in **Section 2**, there are a variety of approaches to this problem, ranging from fixed IP addresses to broadcast discovery protocols such and an expanding ring search (this latter approach does have the attractive property that it is both "ad hoc" and "pure" peer-to-peer). The approach we took, largely to ease implementation within the context of our organization's existing 802.11 infrastructure, is to maintain a central store for all IP addresses, contact list information, and information about online users. The database is kept on a server to allow access either through the device, or through a web interface. Such a database could also be maintained in a peer-to-peer fashion if desired. A Secure Socket Layer (SSL)-based web interface [5] is used for account creation and contact list management.

After a user has logged onto the system, the program executes a retrieve routine for the IP addresses. This routine will then establish a session with the database to read and write data. In order to limit accessibility and address security concerns, the database will maintain its own authentication with client privileges. The device or user interface will then write the IP address to a master table and retrieve the contact list information from a table. Upon completion of these two operations, the session to the database is closed. The database will only be accessed on a need basis and the device is required to establish a connection every time it tries to get information. This stateless transactional model reduces the both the amount of trusted state and the window of vulnerability. With transmission of password and data from the authentication system, encryption is needed to provide security and to maintain data integrity. We used Apache with SSL for the server. SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity, and encryption for privacy.

5.2. Peer-to-Peer Key Exchange

Diffie-Hellman key negotiation [3] is used for the IMP. The resulting shared secret is used as a key for the Blowfish encryption algorithm discussed in the next section. Keys are dynamically created for each conversation. There are no stored keys anywhere in the system.

5.3. Voice Encryption

The encryption software operates in two phases: key exchange and voice encryption. The Diffie-Hellman key negotiation protocol is used for key exchange for a voice session. Encryption of the voice data stream takes place immediately before transport and consists of enciphering the data with the negotiated key for the conversation.

We used the Blowfish [23] cipher with a 128-bit key to encipher the voice data stream. Blowfish has two major advantage for an embedded device: (1) it is compact, running in less than 5 K of memory used; and (2) it uses only simple machine operations such as exclusive OR's, additions, and table lookups. These advantages are significant for a RISC embedded device such as the Elf with limited memory. An AES implementation might also be interesting. As conversation is interactive, performance gains pay off in a reduction of delays from encryption operations performed on the voice streams.

Blowfish is a 64-bit block cipher. The data to be encrypted is split into 64-bit blocks and encrypted by repeated passes through a simple function that is repeated sixteen times. In this it resembles the substitution and confusion schemes [24] used in DES [16] but as we have noted above, Blowfish is very sensitive to the concerns of embedded software implementation.

5.4. Security Evaluation

The security goals of the system were voice privacy and integrity. Voice scrambling has a long history [12], some of this experience showing that voice scrambling is harder than it looks. While we believe that SSL and Blowfish are robust, we wished to validate the system in several ways to test our design and implementation, mainly looking for coding mistakes, e.g., to check that we were not sending cleartext packets or clear speech by accident..

We built a small 802.11b promiscuous listening system with which we could emulate an opponent's presumed capabilities. This program runs on a computer with an 802.11b card that can intercept IP packets on the same LAN. We dumped traces into a log while the system was operating. While MAC and IP addresses and UDP port numbers are visible and might aid traffic analysis, a variety of known techniques such as Onion Routing could be used to create a robust privacy overlay if one were required, so this did not concern us. Other classic attacks on such a system would include attacking the authentication phase of the security protocol. Our use of an SSL interface insulates against problems with the authentication service, and barring bugs in SSL, will be secure. After a visual (but not exhaustive) examination of the tcpdump-formatted log data we could find no useful data.

To study the security of the system for voice privacy, a packet sniffer (and thus a "promiscuous listener" at several layers…) was implemented, and selected packets (those between the two devices using UDP, which we presumed were voice) were captured. The packet format is conventional, UDP encapsulated in IP and IP encapsulated in an Ethernet frame. The three headers (Ethernet, IP, and UDP) are stripped and then the UDP payload, where the voice data is found, is sent through the same functions to decode and play it

as is done in the secure voice over IP software was played out through the speaker.

In every experiment we performed the output of the sniffer was (loud) noise, which based on listening we believe reasonable to declare white noise. A complete study would use an RF analyzer to detect signals and emissions from the device outside of the bands used by 802.11b, and would apply various spectral tests to ensure that the encoded voice was indeed indiscernible from noise. While not trivial, the EM shielding problem is understood, and the IMP's architecture is independent of the choice of cipher we rely on for voice encryption. Relative to the overall architectural goal of the system, it was remarkably easy to provide encrypted voice; it should be more common.

6. Experiences

As noted in **Sections 1** and **7** we have not attempted to experiment with any scalability issues – that would be challenging indeed with the two IMPs we have actually constructed. However, that being said, we have considerable operational experience with these two systems, which have proven a great deal of fun to use when roaming around our organization's campus, one well-equipped with 802.11b access points.

The device's software user interface has proven quite effective, if for no other reason than it resembles conventional voice equipment such as our mobile telephone design target (except, obviously as can be seen in *Figures 1* and *2*, for packaging). The main user interface program is a menu navigation system written in Python. This allowed the user interface to use secure HTTP connections, write to files, run system programs, manipulate strings, and create complex data structures with ease. The code from the LCD library was used as an extension module to the Python library to give us access to the LCD functions from Python. The user is presented with the basic menus of the system; contact list, call arbitrary IP address, and system menu.

The contact list is a scrolling list of names of people that the user has specified that they are interested in talking to. The contact list allows the user to select a name and start either an encrypted or unencrypted to that person. The user is also allowed to refresh the contact list to retrieve the current data. The call arbitrary IP address menu allows the user to type in the IP address of someone they wish to call that is not registered on the contact list. The system menu allows the user to view their network information, signal strength, and log out of the authentication system and device.

The user presents a username and password to log into the authentication system and device to receive service. By having the authentication take place at the central authentication system instead of on the actual IMP device, any user on the system may log in from any IMP. The keypad numbers are mapped to the letters of the alphabet using the encoding used for touch-tone phones. The authentication system submits username and password via an SSL connection, and a contact list for the user is returned if the password is correct. If the user is denied access to the authentication system, the interface is reinitialized to allow additional login attempts.

The most important aspect of the user interface is allowing the user to call another party and talk to them using the SVOIP program. Once a user selects a name to call, the user interface uses the data from the authentication system to resolve the name to an IP address. This address is used to contact the remote party to negotiate a port to be used for the call, as well as whether or not the call will be encrypted. This connection is done using an SSL socket for security. The SVOIP program is then started with the negotiated parameters and the user is presented with the security information on the LCD. The user is also able to cancel the call via the keypad. On the receiving end, the user interface listens to a port that is used to exchange the negotiated parameters. When a connection is received, the user interface is interrupted and information indicating the type of incoming call is displayed. The SVOIP program is started as well, and the receiving user is also allowed to cancel the call via the keypad. When the call is completed, both users are returned to their original location in the navigation menu.

The use of Linux provided the advantage of a familiar programming environment, which accelerated our enhancements. It is hard to overestimate the value of programmability in an end-to-end architecture!

The voice quality is quite good. We did some comparisons among our mobile telephones (this turned out to give us a reasonable selection of providers) with wireless technologies including PCS, GSM and CDMA, and the IMP conversations were comparable and often better in voice quality. In similar comparisons with desk phones using commercial VOIP services, these services often sounded better than the IMP, but often sounded worse. The IMP's voice quality was generally inferior to a POTS telephone.

The effects of signal quality (*e.g.*, when distance from the access point is too great) are to say the least, dramatic. The particular Blowfish mode we have used for the design does not require cryptographic resynchronization, but if such resynchronization was required (for example in the case where a feedback mode of the Blowfish cipher was used) designs are available for DES which we believe are easily reused in the Blowfish context.

While as designers we are rather fond of the prototype systems, it would benefit from the size, weight and other interface improvements of commercial systems, as well as improvements in battery technology (although 2-3 hours of active use is similar to the battery life we experience using commercial cellular telephone systems). In spite of these limitations (and perhaps this is inappropriate to note in a technical paper, but is our experience nonetheless) that our colleagues are initially curious about the systems, and become somewhat envious of the Internet Mobile Phones once their function is understood.

7. Conclusions

7.1. Prior and Related Work

Architectural work by DeTreville and Sincoskie [2] described a substantial experiment with voice over Ethernet. In that experiment, custom handsets were constructed and attached to a 3Mbps Ethernet to build a packet-based PBX. One of the authors has experience with these phones, which performed remarkably well. This system demonstrated clearly that a packet-based voice telephony system could be realized and might have a variety of attractions. Using the access network versus bearer service distinction of **Section 1**, this system can be seen as unifying the access and bearer service, as the IMP does in the wireless domain. The agenda for this research is discussed in the recent article by Sincoskie [26].

Similar LAN telephony systems were implemented by a group at Xerox PARC [27,30], which attempted greater integration with a desktop workstation environment; other researchers used the same LAN for voice and data packets and like the PARC team attempted to merge desktop functionality with telephony services [15,19,20,22]. The IMP's only role is to be a secure mobile phone with IP access and bearer services.

Commercial VOIP service is widely available (e.g., from Vonage and Net2Phone), but the service is largely between switches internal to the telephony system rather than directly between end devices. Some switch vendors are successfully selling VOIP phones to commercial customers who wish to maintain a single network infrastructure, letting the vendor's switch interface to the telephone company, acting as a PBX.. Nextel uses a packet overlay on telephony-style channels to provide services such as a "Walkie-Talkie" mode and group services, but does not leverage the Internet in the way we have with the wireless LAN technology used by the IMP.

The ICEBERG [18,29] project at Berkeley builds a signaling and service infrastructure using advanced cluster computing technology and a novel access point architecture. The IAP access points are used to interconnect mobile devices (GSM telephones and 802.11 laptops are described in the papers) to the Internet, and provide signaling and call setup services to the mobile user which seem superior to H.323 and SIP[9] in the ICEBERG team's analysis. Our view is that the ICEBERG system is a powerful service overlay for IP, and if one chose to use the IMP as a mobile device for ICEBERG, it would simply replace some of its present software with interface software for the IAP.

While consideration has been given to various forms of Internet appliance (see, for example the useful framework in [6]), as well as handheld computing (e.g., *Itsy* [8] which also addresses power and packaging issues) we believe our secure wireless LAN-based mobile phone system is unique.

The most similar mobile device functionality was achieved by Shih, et al. [25] using a PDA-based system based on the Compaq iPAQ platform, with 802.11b communications. They studied a novel scheme for power management, and added a "miniBrick" device interface designed to augment the 802.11 LAN. In the infrastructure mode of operation described in their paper, the device does not operate with arbitrary 802.11 infrastructures (although it appears it could operate peer-to-peer), and the focus was neither the programming environment nor the security. Their goal was improved power management for the mobile device rather than a reexamination of network architecture implied by an IP-based mobile telephony infrastructure. The IMP is possibly the simplest possible proof-of-concept experiment for an all-IP mobile telephony infrastructure.

7.2. Contributions

The IMP can be built by a hobbyist, which we believe has been aided by the technical detail in selected portions of this paper. While the prototype IMP realization is clumsy relative to devices such as commercial mobile phones and pagers, it could be far more portable if commercially packaged. A desirable consequence of our work would be the development of an ad-hoc user community of IMP users; we have GPLed our software and contributed it back into the Linux mainstream, so our software efforts need not be duplicated.

We believe the important contributions of our work are three:

1. We have shown (albeit at a very small scale) that an Internet-based device can use a wireless LAN link to emulate most of the desirable properties of cellular telephony, while remaining IP-based. While it was in no way designed to interface to the conventional telephone system, the advent of telephony/VOIP gateways might create interesting opportunities for interoperability experiments.
2. We have shown that in spite of using wireless LAN technology, strong encryption can be used over the IP layer to secure the channel using modern algorithms for cryptography and key negotiation. The ease with which this was accomplished should be embarrassing to cellular telephone vendors.
3. We have implemented the system in a "peer-to-peer" fashion (with the exception of our directory scheme), with the explicit goal of providing an interface to wireless LANs which maximizes Internet interoperability and minimizes the need for centralized administration.

7.3. Future Work

The Internet Mobile Phone prototypes described in the paper suggest that interesting ad-hoc voice networks could be constructed with wireless access points, for example those that are deployed throughout our organization's facilities – some individuals and organizations are now permitting public access to these access points, albeit with some of the limitations such

as NATS and firewalls mentioned in **Section 1** While 802.11b contention might provide a limitation on multiplexing, the fixed bandwidth required for voice encoding suggests that capacity upgrades applied locally, such as switching to 802.11a or 802.11g, could upgrade service "hot-spots" as required. For the devices to interface with commercial VOIP offerings, they would simply have to use the same voice coding; the security features could be selectively deployed as they are with the IMPs.

Extensibility research would examine how easily features are added to these P2P "phones" based on an end-to-end architecture as compared to GPRS and other approaches being attempted by the telephone community. The straightforward addition of high-grade security to our system suggests an architectural advantage. Useful extensions to achieve greater Internet interoperability would include implementations of SIP[9] and IPSEC [15]..

Acknowledgements

Chuck Davin, Jon Moore, Ashutosh Dutta and John Ioannidis provided useful commentary and constructive criticism. Portions of the work were supported by DARPA under Contract F39502-99-1-0512-MODP001.

REFERENCES

[1] J. N. Daigle and J. D. Langford, "Models for analysis of packet voice communications systems", IEEE JSAC, Sept. 1986, pp. 847-855.

[2] DeTreville, J. and W. D. Sincoskie, "A Distributed Experimental Communications System", IEEE JSAC, Vol SAC-1, No. 6, Dec. 1983.

[3] Diffie, W. and Hellman, M. "New Directions in Cryptography", IEEE Trans. Info. Theory, Nov., 1976.

[4] W. Dorsey *et al*. "Nautilus Secure Phone Homepage", http://www.lila.com/nautilus/

[5] R. S. Engleschall, "User Manual mod_ssl version 2.8." Jan. 2001. http://www.modssl.org

[6] S. Gillett, W. Lehr, J. Wroclawski and D. Clark, "Do Appliances Threaten Internet Innovation?" IEEE Communications, October 2001.

[7] A. Gopal, I. Gopal and S. Kutten, "Hardware Flooding", Proc. SIGCOMM, September 1991

[8] W. R. Hamburgen, et al., "Itsy: Stretching the Bounds of Mobile Computing", IEEE Computer 34(4), 2001, pp. 28-37.

[9] M. Handley, et al, "SIP: Session Initiation Protocol", IETF RFC 2543, March 1999.

[10] Inhand Electronics, Inc. "Elf (Rev B) Development Platform User Guide" 2001

[11] J. Ioannidis, D. Duchamp and G. Q. Maguire, Jr., "IP-based Protocols for Mobile Internetworking", Proc. SIGCOMM, Sept. 1991, pp. 235-245

[12] D. Kahn, "The Code Breakers", Macmillan (New York), 1967.

[13] R. King, "The Arm Linux Project." March, 2002. http://www.arm.linux.org.uk/

[14] Matrix Orbital Corporation, "LK204-25 User Manual". 2000.

[15] N. Maxemchuk, "An Experimental Speech Storage and Editing Facility", BSTJ, Oct. 1980, pp. 1383-1395.

[16] Meyer, C. and Matyas, S. "Cryptography: A New Dimension in Computer Data Security", Wiley (New York), 1982.

[17] Phillips Corporation. "UCB1200 Advanced Modem/audio Analog Front end Product Specification" Jul. 1998.

[18] B. Raman, H. J. Wang, J. Shih, A. D. Joseph and R. H. Katz, "The Iceberg Project: Defining the IP and Telecom Intersection", in IEEE IT Pro, November/December 1999, pp. 22-29

[19] P. V. Rangan, H. Vin and S. Ramanathan, "Communication architectures and algorithms for Media Mixing in Multimedia Conferences", in IEEE/ACM Trans. Net. 1(1), 1993, pp. 20-30.

[20] A. Ruiz, "Voice and Telephony Applications for the Office Workstation", in Proc. 1st Intl. Conf. On Computer Workstations, November 1985, pp. 158-163.

[21] J. Saltzer, D. Reed and D. Clark, "End-to-End Arguments in System Design", in ACM TOCS 2(4), Nov. 1984, pp. 277-288.

[22] C. Schmandt and M. McKenna, "An Audio and Telephone Server for Multi-Media Workstations", Proc. 2nd IEEE Conference on Computer Workstations, March 1988, pp. 150-160.

[23] B. Schneier. *Applied Cryptography*. New York, NY: John Wiley & Sons Inc; 1996.

[24] Shannon, C. "Communication Theory of Secrecy Systems", BSTJ, No.4, 1949.

[25] E. Shih, P. Bahl and M. J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices", Proc. MOBICOM '02, Atlanta, GA, September '02.

[26] W. D. Sincoskie, "Broadband Packet Switching: A Personal Perspective", IEEE Communications Magazine, July 2002, 40(7), pp. 54-66.

[27] D. Terry and D. Swinehart, "Managing Stored Voice in the Etherphone System", ACM TOCS, 6(1), February 1988, pp. 3-27.

[28] R. Thayer, N. Doraswamy and R. Glenn, "IP Security Document Roadmap", *Internet RFC 2411*, http://www.ietf.org/

[29] H. J. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. S. Shih, L. Subramanian, B. Z. Zhao, A. D. Joseph and R. H. Katz, "ICEBERG: An Internet-core Network Architecture for Integrated Communications", IEEE Pers. Comms., 2000.

[30] P. T. Zellweger, D. B. Terry and D. C. Swinehart, An Overview of the Etherphone Systems and its Applications", Proc. 2nd IEEE Conf. On Computer Workstations, March 1988, pp. 160-168.