# Survivable Mobile Operating System

Mohammad Samarah and James Whittaker
The Center for Software Engineering Research
Florida Institute of Technology
150 West University Boulevard, Melbourne, FL 32901-6988
Phone: (321) 674-8763
Fax: (321) 674-8192
Email: msamarah@zach.fit.edu and jw@se.fit.edu
Contact Author: msamarah@zach.fit.edu

Florida Tech Technical Report CS-2001-5

## Abstract

Although there is a large body of work on cryptographic techniques and algorithms that provide basic building blocks to solve specific security problems, relatively little work has been done in investigating security issues in mobile system contexts. Conventional security controls work well for static code, but breakdown with code mobility. In this paper, we investigate the need for end-to-end security in mobile and wireless operating systems. We study the implication of mobility – specifically, ways in which the operating system may facilitate communication security. We suggest a framework for designing security into mobile devices by building encryption into the mobile device thus providing end-to-end security and eliminating carrier-provided encryption overhead.

**Keywords**: Mobile operating systems security, wireless protocols, mobile agents, communication languages and protocols.

# Survivable Mobile Operating System

Mohammad Samarah and James Whittaker
The Center for Software Engineering Research
Florida Institute of Technology
150 West University Boulevard, Melbourne, FL 32901-6988
Phone: (321) 674-8763
Fax: (321) 674-8192
Email: msamarah@zach.fit.edu and jw@se.fit.edu
Contact Author: msamarah@zach.fit.edu

Florida Tech Technical Report CS-2001-5

**Abstract -** Although there is a large body of work on cryptographic techniques and algorithms that provide basic building blocks to solve specific security problems, relatively little work has been done in investigating security issues in mobile system contexts. Conventional security controls work well for static code, but breakdown with code mobility. In this paper, we investigate the need for end-to-end security in mobile and wireless operating systems. We study the implication of mobility – specifically, ways in which the operating system may facilitate communication security. We suggest a framework for designing security into mobile devices by building encryption into the mobile device thus providing end-to-end security and eliminating carrier-provided encryption overhead.

**Keywords**: Mobile operating systems security, wireless protocols, mobile agents, communication languages and protocols.

## 1 Introduction

The far-reaching influence of the Internet has resulted in an increased interest in embedded real-time mobile operating systems, which are poised to play a key role in the implementation of successful Internet, wireless, and electronic appliances based applications in the future. While there is still considerable hype concerning mobile and wireless technologies, there is also an increasing awareness of the problems involved. In particular, that these applications will not be successful unless security issues can be adequately handled.

Many researchers in the mobile computing arena share the same vision: ubiquitous access to information, data, and applications. Ubiquitous access refers to the ability of users to access these computing resources from almost any terminal.

Recent developments relating to the Internet are establishing solid foundations for wide-area ubiquitous computing systems. The recent development of cross-platform languages, the most important of which is Java, enable the deployment of computing systems that span multiple computing platforms and communication networks. Java provides a means of building applications that are accessible from virtually any Internet-connected terminal. This is true because the only requirement for running Java programs or applets is a Java-enabled Web browser, that is a standard component on most computer desktops.

Further evolution of Internet technologies will yield a wide-area network based on component-oriented, dynamic applications, which support efficient, scalable resource sharing for a large number of mobile and nomadic users. As users gradually grow to rely on the Internet as an indispensable tool, most users will become mobile or nomadic users, or both. While mobile users access the Internet from a portable computer, nomadic users may move from terminal to terminal. In either case, a user would ideally be able to accomplish the same tasks with equal ease from any location either on his portable computer or at any Internet-connected terminal.

Although there is a large body of work on cryptographic techniques and algorithms that provide basic building blocks to solve specific security problems, relatively little work has been done in investigating security issues in mobile system contexts. The introduction of mobile code significantly increases the risks involved in Internet and WWW-based applications. For example, if we allow mobile code to enter a private network, we must offer a platform so that it can execute correctly while ensuring that it will not have harmful effects on our hosts or any other processes in the network. When sending out mobile code, we should also be able to guarantee specific aspects of their behavior. We are not only interested in whether it carries out their intended task correctly, but also need to be able to provide the ability to defend itself against attacks from other mobile code or to survive in a potentially malicious environment. It needs to be properly protected by secure hardware and/or cryptographic mechanisms that allow the mobile code to defend itself against attacks by hosts or other mobile code.

In this paper, we study mobile and wireless security protocols and propose a framework for end-to-end security. In particular the ability of mobile code to traverse external networks and securely execute on a remote device running an embedded real-time operating system.

The remainder of the paper is structured as follows. In Section 2, we present the dimensions that must be considered when designing mobile security. Section 3 provides an overview of wireless and mobile security. Section 4 studies mobile code security. Section 5 concludes the paper and describes related work.

## 2 Designing Mobile Security

There are several dimensions that must be considered when designing security in mobile computing networks and applications:

### 2.1 Physical Security, Policies and Procedures

There is no point in implementing expensive advanced security systems while the physical security of end user devices, base stations, and information servers is ignored. A notebook left in the back seat of an unlocked car is an obvious and often common security violation.

This potential problem will soon be exacerbated with the advent of inexpensive Personal Communication Service/Personal Communication Network (PCS/PCN) micro-cells located in small and unattended sites throughout user communities.

### 2.2 Application and System Security

The use of user passwords and similar mechanisms is a very common method of ensuring security. Security designers should employ advance authentication techniques such as biometrics and other hardware-based techniques.

### 2.3 Firewalls — Security Gatekeepers

A firewall is a network device designed to prevent unauthorized access to private networks. Firewalls are implemented in hardware, or software, or both and frequently are used to prevent unauthorized users from accessing private networks connected to the Internet. All packets entering or leaving the private network pass through the firewall, which examines each one and blocks those packets that do not meet the private network security criteria.

There are several types of firewall techniques used in practice:

1. Packet filter: Examines each packet entering or leaving the network and accepts or rejects it based on user-defined security criteria. Packet filtering is very effective and transparent to the network users, however it is difficult to configure and is susceptible to IP spoofing.
2. Application gateway: Applies security mechanisms to specific applications, such as FTP and Telnet traffic. This is a very effective measure, but may lead to performance degradation.
3. Transport-level gateway: Applies security mechanisms to TCP and UDP connections. Once the connection has been established, packets can flow between the hosts without further intervention from the firewall.
4. Proxy server: Intercepts all packets entering and leaving the network. The proxy server effectively hides the internal network to all other external networks.

Firewalls are the first line of defense in protecting private networks. In practice, firewalls use two or more of these techniques.

### 2.4 Encryption

Encryption involves scrambling digital information with complex mathematical algorithms and is the most effective protection available against security intrusions into wireless and wire line communications.

Many cellular carriers are now providing encryption between cell sites and the mobile telephone switching office (MTSO). However, the last segment between the end user device and the base station is not encrypted and this is where potential security breaches may occur. To deliver end-to-end security, encryption/decryption capabilities must be built into the end user device itself.

There are three types of keys used in encrypting data:
1. A private key known only by the sender and the recipient
2. A private/public key combination
3. A one-time key

In private-key systems, the two parties have a secret key with which they use to encrypt and decrypt data. The private/public key combination is more secure, however. In this scheme, the recipient's public key — available to all that need it to send encrypted data — is used to encode information for transmission. The recipient uses a private key associated with the set to decode the information.  The one-time key method is based on the generation of a new key every time data is transmitted. A single-use key is transmitted in a secure mode and once used, becomes invalid.

### 2.5 Electronic Signatures

Electronic signatures can be used to ensure that users are who they claim to be. With the appropriate hardware and software — a system can request a valid signature before the user is granted access. While the primary use of such software is in financial and banking applications, it can be also used as a substitute for password authentication.

### 2.6 End-To-End Encryption

While each of the security schemes discussed earlier provides a certain amount of security in and of itself, we believe the best scheme is one based on end-to-end encryption using asymmetric cryptography, where not even the network carrier control center has access to data being transferred. To achieve this, the client machine and the information server must each perform encryption/decryption as appropriate, depending on the direction of the transmission.

This approach is independent of any security that the network provides. In fact, depending on the number of mobile users involved, the cost of carrier-provided encryption may be much higher than end-to-end encryption implemented on the user device.

## 3 Mobile and Wireless Security

Wireless security is not much different from wired security. To provide a secure environment, wired or not, we must authenticate whom we are talking to, secure the data as it travels from the handheld device to the destination host, and ensure that the traffic hasn't been altered en route.

However, wireless has some unique difficulties, such as limited bandwidth, high latency and unstable connections. Several options exist to address these issues: Wireless Transport Layer Security (WTLS) ? a Secure Sockets Layer-like (SSL) security protocol ? and connection-oriented security communication using established protocols, such as Mobile IP security extensions, IP Security (IPsec) and Secure Shell (SSH).

In this section we study mobile and wireless security, in particular how to apply conventional security and authentication protocols in a mobile setting environment. We start by providing an overview of Mobile IP and the Wireless Application Protocol (WAP).

### 3.1 Mobile IP

Mobile IP is an open standard (RFC 2002) specified by the Internet Engineering Task Force (IETF) in 1996. It is an extension to the standard Internet Protocol (IP) that enables users to maintain connectivity while roaming between IP networks. Mobile IP enables devices like patient monitors, voice-over IP phones, and computer laptops to communicate seamlessly while traveling between different locations or networks. In an era when maintaining continuous connectivity while roaming has become increasingly important to everyday life, it has become even more crucial for mission-critical fields such as medical services.

Addressing and routing in IP networks are fixed at locations, so a device on a network is reachable because it has an address on the network. IP networks encounter the challenge that when a device is no longer associated with its former IP address on its home network, its active sessions are dropped. Mobile IP was created to enable users to retain the same IP address while traveling to a different network, thus ensuring that a roaming individual could continue communications without dropping open sessions and connections.

Mobile IP is comprised of three entities: The Mobile Node (MN), the Home Agent (HA), and the Foreign

Agent (FA). A Mobile Node is a device capable of performing network roaming. Examples of Mobile IP clients are cell phones, PDAs, or laptops whose software enables roaming capabilities. The Home Agent is a router on the home network serving as the anchor point for communication with the Mobile Node; it tunnels packets to the roaming Mobile Node. The Foreign Agent is a router that functions as the Mobile Node's point of attachment when it travels to a foreign network, delivering packets from the Home Agent to the Mobile Node.

Mobile IP employs the Care-of Address and Correspondent Node. The Care-of Address is the termination point of the tunnel toward the Mobile Node when it is not on its home network, while the Correspondent Node is the device that the Mobile Node is communicating with, such as a Web server.

## 3.2 WAP: The Wireless Application Protocol

The Wireless Application Protocol (WAP) is an application environment and a set of communication protocols for wireless devices designed to enable manufacturer, vendor, and technology independent access to the Internet and advanced telephony services.

Ericsson, Nokia, Motorola, and Unwired Planet founded the WAP Forum in the summer of 1997 with the initial purpose of defining an industry-wide specification for developing applications over wireless communications networks. The WAP Forum now includes more than 500 telecommunications and software companies who saw the need to cooperate to create a wireless application protocol [13].

The WAP specifications define a set of protocols in application, session, transaction, security, and transport layers. WAP utilizes Internet standards such as XML, User Datagram Protocol (UDP), and Internet Protocol (IP). Many of the protocols are based on Internet standards such as Hypertext Transfer Protocol (HTTP) and Transport Layer Security (TLS) but have been optimized for the unique constraints of the wireless environment: low bandwidth, high latency, and less connection stability.

Internet standards such as Hypertext Markup Language (HTML), HTTP, TLS and Transmission Control Protocol (TCP) are inefficient over mobile networks, requiring large amounts of text-based data to be sent. Standard HTML content cannot be effectively displayed on the small-size screens of handheld PCs, pocket-sized mobile phones and pagers.

WAP utilizes binary transmission for greater compression of data and is optimized for long latency and low bandwidth. WAP sessions cope with intermittent coverage and can operate over a wide variety of wireless transports. The Wireless Markup Language (WML) and Wireless Markup Language Script (WMLScript) are used to produce WAP content. They make optimum use of small displays, and navigation is optimized for mobile input devices. WAP content is scalable from a two-line text display on a basic device to a full graphic screen on more capable mobile devices.

WAP is based on a layered architecture. The lightweight WAP protocol stack is designed to minimize the required bandwidth and maximize the number of wireless network types that can deliver WAP content.

## 3.3 Mobile IP Security Model

A mobile host is by nature more vulnerable as far as information security is concerned. Mobile IP is designed to support a range of security models, ranging from no security to weak security to strong security. It may use public/private keys, trusted hosts, and the IPsec protocol.

The most basic security scheme lies in the use of shared secret keys. If the Security Parameter Index for any two nodes (i.e., an MN and an HA) specifies mutual authentication using shared secret keys, the following exchange is one possible scenario that could occur as shown in Figure 1 [10]:

1. The MN chooses a random number, $R_1$, and asks the HA to encrypt it using a shared key, $K_{MN\text{-}HA}$.
2. The HA sends the MN the encrypted $R_1$, denoted as $E\{K_{MN\text{-}HA}, R_1\}$, and also chooses a random number, $R_2$, which it sends to the MN.
3. The MN decrypts the HA's message and verifies $R_1$. The MN also encrypts the HA's random number, $R_2$, and sends the encrypted random number, $E\{K_{MN\text{-}HA}, R_2\}$, to the HA.

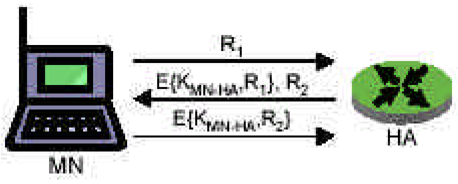


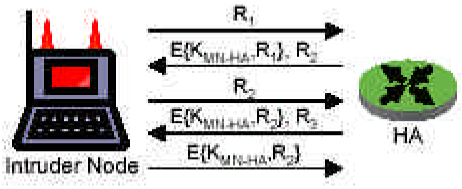
**Figure 1 - Shared Secret Key**

**Reflection Attacks**

However, a subtle security flaw makes this authentication technique susceptible to a form of replay attack [5]. An Intruder Node can conduct this attack by sending authentication request messages to the HA and replaying them back to the HA in a certain sequence to break the authentication. This attack, known as the reflection attack, is conducted as follows:

1. The Intruder Node chooses a random number, $R_1$, and asks the HA to encrypt it.
2. The HA sends the Intruder Node the encrypted random number, $E\{K_{MN\text{-}HA}, R_1\}$, and also chooses a random number, $R_2$, which it sends to the Intruder Node.
3. Since the Intruder Node does not know $K_{MN\text{-}HA}$, it cannot encrypt $R_2$ and respond to the HA's challenge. However, the Intruder Node can open another session with the HA or another HA using the same secret key. In the second session, the Intruder Node replays $R_2$ to the HA, hoping to trick the HA into encrypting $R_2$.
4. The HA encrypts $R_2$ and sends the encrypted random number, $E\{K_{MN\text{-}HA}, R_2\}$ back to the Intruder Node.
5. The Intruder Node then replays $E\{K_{MN\text{-}HA}, R_2\}$ for the first session to complete the authentication process with the HA.

Reflection attacks can be countered by ensuring that valid nodes do not perform the exact same procedure for mutual authentication. If the HA authenticates using a different key or encryption scheme from the MN, an Intruder Node cannot trick the HA into generating the correct response for the HA's challenge.



$R_1$
$E\{K_{MN\text{-}HA},R_1\}, R_2$
$R_2$
$E\{K_{MN\text{-}HA},R_2\}, R_3$
$E\{K_{MN\text{-}HA},R_2\}$
Intruder Node
HA

**Figure 2 - Reflection Attack**

**Reflection Attack Countermeasures**

Reflection attacks can also be countered by having initiating nodes authenticate first. If an Intruder Node has to authenticate first, it would fail authentication.

An example of this exchange would be as follows:

1. The MN initiates communications with the HA.
2. The HA chooses a random number, $R_1$, and asks the MN to encrypt it using a shared key, $K_{MN\text{-}HA}$.
3. The MN sends the HA the encrypted $R_1$, denoted as $E\{K_{MN\text{-}HA}, R_1\}$, and also chooses a random number, $R_2$, which it sends to the HA.
4. The HA decrypts the MN's message and verifies $R_1$. The HA also encrypts the MN's random number, $R_2$, and sends the encrypted random number, $E\{K_{MN\text{-}HA}, R_2\}$, to the MN.

If an Intruder Node attempts to register as a valid MN, the Intruder cannot perform step 3 because it cannot encrypt $R_1$. An Intruder could still thwart this scheme if somehow the Intruder were able to pose as the HA and trick the MN to initiate a registration request with the Intruder instead of the HA. While this trick is possible, it is considerably more difficult to accomplish than posing as a legitimate MN.

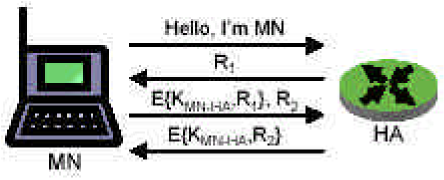


**Figure 3 - Reflection attack counter measure**

**Shared Secret Key with Hashes**

While shared secret keys can provide strong authentication services, none of the schemes described above provides message integrity for registration requests and replies. Message integrity can be achieved by encrypting the entire message and sending the encrypted message along with the original message. The receiver would then decrypt the encrypted message and compare it with the original message to ensure that no changes were made during transit. Unfortunately, this method of ensuring message integrity requires more than double the necessary bandwidth for conducting registration since the encrypted message is generally larger than the original message. Hashing algorithms can be used to avoid this overhead and achieve similar results.

Hashing algorithms can assure the integrity of messages sent between the MN and the HA. A number of different hashing algorithms are available (e.g., NIST's SHA-1); however, for Mobile IP [8], the default authentication algorithm uses keyed Message Digest 5 (MD5) [9] as the hashing algorithm. MD5 processes a message of any arbitrary length and produces a 128-bit message digest of the original message. MD5 is computationally inexpensive and relatively simple to implement.

For Mobile IP registration using the default settings, the MN computes an MD5 hash based on the shared secret key and the registration request message. The MN sends both the hash and the message to the HA, which then performs the same MD5 hashing algorithm to the message and compares the hash with the MN's hash. If the hashes match, then the HA is assured of the MN's identity.

### Message Extension Attacks

However, the standard MD5 hash algorithm is susceptible to message extension attacks [11]. This attack is possible because of the iterative design of MD5, which allows an attacker to add to the original message and forge a valid hash by reapplying the MD5 algorithm to the new extended message. This attack can be thwarted by inserting the shared secret key before and after the message to be transmitted (prefix + suffix mode). However, very short messages may still be vulnerable [4]. This can be avoided by adding a sufficient amount of padding after the prefix and before the message [8]. Unfortunately, padding is not specified in the default authentication algorithm for Mobile IP. If an attacker is successful in appending additional information to the original message, it is uncertain how the receiving node would treat the appended information. Since the original IP header is not being modified, the HLen and Length fields would remain the same and should tell the receiver where the packet ends. At best, the receiver may simply disregard the additional information. Alternatively, the receiver could treat it as a new packet, or worse, it could place the information in a buffer waiting to be overflowed and exploited.

### Diffie-Hellman Key Exchange

If two nodes wishing to authenticate do not already have an existing security association, the nodes can perform the Diffie-Hellman key exchange algorithm [2] to establish a secret key for registration. The basic algorithm is performed using the following steps as shown in Figure 4 [10]:

1. Both nodes agree to use a public prime number and generator number.
2. Each node generates a random, secret value, which it keeps private.
3. Each node computes a public value and sends this value to the other node. The public value is derived mathematically from the random value, the prime number, and the generator number.
4. Each node mathematically combines the public value received from the other node with its own random, secret value to obtain the shared secret key. Upon completion of the algorithm, both nodes have a shared secret key with which they can perform the secure communication.

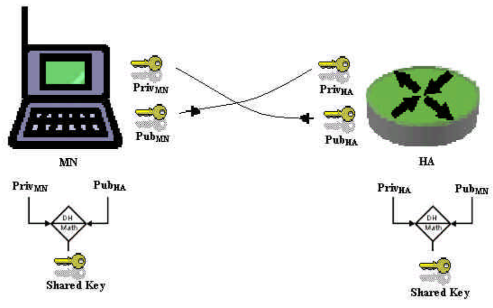


**Figure 4 - Diffie-Hellman Key Exchange**

### Man/Woman-in-the-Middle Attack

However, the basic Diffie-Hellman algorithm is vulnerable to an active attack known as the man or woman-in-the-middle attack. An Intruder could intercept each legitimate node's information exchange and establish separate secret keys with each legitimate node. The Intruder can then relay information between the two legitimate nodes and make it appear as if the two legitimate nodes are talking directly with each other. This direct attack can be avoided by having one node, such as the HA, retain public numbers that are published and cannot be modified by the Intruder. In this way, legitimate nodes wishing to establish secret keys with the HA will know that they are communicating directly with the HA as long as the published numbers are used in the Diffie-Hellman algorithm.

### Timing Attack

The Diffie-Hellman algorithm has also been proven to be vulnerable to timing attacks that could potentially reveal the shared secret key [6]. Attackers could identify the random, secret value by measuring the time required to perform the Diffie-Hellman algorithm. The timing attack is made easier when the random value is static and does not change from operation to operation. The timing attacks can be countered as well by masking the timing characteristics and preventing attackers from obtaining information necessary to conduct a timing attack. Additional information on these countermeasures can be found in [6]. Since the Diffie-Hellman algorithm is computationally expensive, the Diffie-Hellman key exchange should be reserved only for authentication between HAs and FAs. As long as the MH and the HA authenticate using their shared secret key, the MH-FA authentication can be inferred without any significant loss in security [7].

**Public Key**

The base Mobile IP specification relies on the use of shared secret keys for authentication, message integrity, and encryption. However, the scalability of key distribution and management is a major problem with the use of secret keys. As a result, many new proposals are being made to add extensions to the base Mobile IP specification to allow the use of public keys [1, 3, and 7]. With trusted third parties, the distribution and management of public keys can be automated and is much more scalable. This Public Key Infrastructure (PKI) makes the task of creating, storing, and distributing keys more manageable and can set the foundation for wide-scale deployment of Mobile IP.

Authentication using public keys is similar to authentication using secret keys. However, instead of using a common secret key for both encryption and decryption, public/private key pairs are used to encrypt or decrypt messages. In addition, different key pairs are used based on which node is being authenticated. For example, the MN authenticates with the HA using the MN's public/private key pair. The MN encrypts a message using its private key and sends it to the HA. The HA can obtain the MN's public key through a directory service, through secure Domain Name System (DNS) based X.509 PKI [12], or through the MN itself. The HA can verify the authenticity of the MN's public key by checking it with a trusted third party that has signed the MN's public key. Once the HA is confident that it has the MN's public key, it can decrypt the MN's message. By successfully decrypting the MN's message, the HA can be assured that the message originated from the MN. The same procedure is mirrored for authenticating the HA with the MN. The HA encrypts a message using its private key and the MN authenticates the HA once it has validated the HA's public key and decrypted the HA's message.

Attackers would have an extremely difficult time breaking this form of authentication for a number of reasons:

1. The attacker has to authenticate with the HA first. Since the attacker would not have access to the MN's private key, the attacker cannot encrypt a message that can be decrypted with the MN's public key. An attacker could send its own public key to the HA and impersonate as the MN. However, a simple check against a trusted third party would show that the attacker is not the legitimate MN.
2. Different keys are used for each transaction. Therefore, an attacker could not perform a reflection attack even if the attacker did not have to authenticate first.

The use of public keys is particularly beneficial in the authentication process with the FA. The base Mobile IP specifications did not require authentication between the MN and the FA or the FA and the HA, because key distribution is difficult in the absence of a network key management protocol [8].

Hashing algorithms such as MD5 can be used in conjunction with public keys. Furthermore, the security of the hashing algorithm can be improved by encrypting the message digest with the private key to create a digital signature and provide additional security mechanisms that can be used to validate the authenticity and integrity of the message transmission. One drawback of public key encryption is that it is computationally expensive and could be a significant processing burden for the MNs. In [7], the authors recommend using public key encryption only for the authentication between the FA and the HA. The existing security associations between the MN and the HA would continue to be leveraged by performing the MN-HA authentication using their shared secret key.

**3.4 WAP Security Model**

The WAP Security model relies on the Wireless Transport Layer Security (WTLS) and Secure Sockets Layer (SSL). The central component in this model is the WAP Gateway, a virtual gatekeeper between the worlds of WTLS and SSL as shown in Figure 5.

A wireless phone communicates with the WAP gateway over a wireless network, using WTLS. The WAP gateway then communicates with the Web server over the Internet, using SSL. WTLS is built on the Internet Security Model.

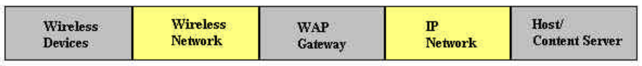


**Figure 5 - The WAP Model**

In the early 1990s the research community experienced a push for stronger Internet security. As a result, the Secure Sockets Layer (SSL) was developed.
A typical SSL security exchange is as follows:

1. The Web browser requests a secure conversation with a Web server.

2. The server provides the browser with its server certificate.
3. The browser authenticates the server by confirming that a valid certificate authority issued the certificate.
4. The browser uses the public key stored in the certificate to encrypt a shared secret key.
5. The browser sends the encrypted shared secret key to the server.
6. The rest of the conversation is encrypted using the shared secret key.

Some web servers require a client certificate, but usually, a server relies on a simple username and password for authentication and non-repudiation.
The Internet Security Model forms the basis for WTLS.

**Wireless Transport Layer Security (WTLS)**

The Wireless Transport Layer Security (WTLS) was formulated specifically to enable secure transactions, yet avoid the power and memory-hungry security solutions used on the Web. It does this by minimizing protocol overhead, utilizing better compression, and employing more efficient cryptography, such as RSA (RC5) or Elliptical Curve Cryptography (ECC) [14].

The kernel of WTLS security is the Wireless Identity Module (WIM). The WIM performs optimized cryptography during handshake, especially for client authentication, and forges long-term, secure WTLS connections.

WTLS came out of TLS 1.0, the Internet standard security protocol. TLS 1.0 is based on SSL 3.0. WTLS goes above and beyond TLS 1.0, offering such features as datagram support, dynamic key refreshing, and optimized handshake.

WTLS provides for client or server authentication and allows for encryption based on negotiated parameters between the handheld device and the WAP gateway. WTLS key exchange protocol is uniquely suited for wireless applications. Three classes of authentication types are available:
1. Class 1 - Anonymous authentication: This model is of limited use and is mainly used for testing purposes since the client has no mechanism to determine the authenticity of the server. The client forms an encrypted connection with an unknown server.
2. Class 2 - Server authentication: This is the most common model used. As with SSL, once clients are assured they are talking securely to the correct server, they can authenticate using alternative means such as user name and password.
3. Class 3 - Server and client authentication: This is the strongest model, as the server and the client authenticate each others WTLS certificate.

Client certificates required for Class 3 authentication pose special management problems. Not only must the key pairs be generated on the mobile device or generated in advance and securely preloaded onto the mobile devices, but also the client certificate has to be safeguarded and managed until the certificate expires. Client certificates need not be retained on the handheld device. Instead, during negotiation, the client may refer the WTLS gateway to a directory saving the bandwidth needed to send the client certificate over the air and improving negotiation performance; however, the WAP gateway needs to trust the directory the client refers to in order to have any assurance of authentication. The directory that holds user certificates also must be available at all times; otherwise the authentication process fails. The key pair associated with the client certificate resides only on the client.

Unlike SSL, WTLS does not provide for end-to-end security. End-to-end security means that the client and server have a secure session, and no other servers intervene. When the Web browser sets up an SSL session with a Web server, the browser and the Web server are communicating directly.

WTLS is not SSL, so it can't directly communicate with SSL-enabled Web servers. An IETF draft in the Transport Layer Security working group aims at adding WTLS extensions to the TLS. The WAP gateway, which translates data packets from the WAP transport protocol (WDP or UDP) to TCP/HTTP, terminates WTLS sessions and initiates SSL sessions to the destination. It is here, at the WTLS gateway that the potential problem exists. Between the time the data is decrypted and decapsulated from WTLS and WAP and re-encapsulated and re-encrypted in SSL, the protected data is exposed ? albeit for only a few hundred milliseconds. To provide for total end-to-end security, encryption must be built into the device as shown in the figure 7. WDP packets are tunneled into an SSL packet and at the destination are decrypted using SSL and WTLS.

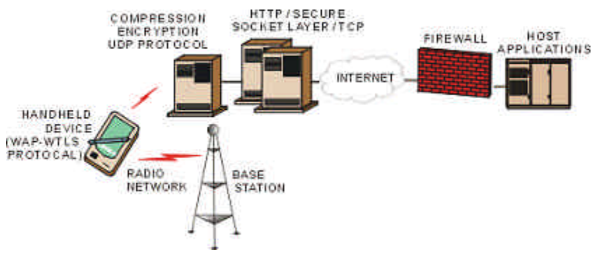


**Figure 7 - End-To-End Security**

## 5 Mobile Code Security

Conventional security controls work well for static code, but breakdown with code mobility. The basic dilemma is that mobile code needs to travel and work autonomously outside trusted domains, but is difficult to protect from unknown host platforms. The counter dilemma is that when mobile code hops between platforms, the receiving platform cannot determine whether tampering has occurred, and the mobile code itself cannot determine whether the platform is malicious.

In this section, we discuss security threats and attacks on mobile code and novel methods to counter them.

### 5.1 Security threats and attacks

Attacks may come from the platform, the mobile code itself, other mobile code, and outside entities. The incoming mobile code may try to gain access to information, corrupt or terminate receiving host platform, consume resources, or deny services to other mobile code. The receiving host platform may try to extract information, corrupt or modify mobile code, deny it services, or terminate it. Other mobile code processes may try to falsify transactions, eavesdrop upon, or interfere with the mobile code activities. Outside entities may try to intercept information, interfere, or subvert the receiving host framework.

Other security threats include disclosure and eavesdropping, alteration and corruption of data the mobile code is carrying, copy and reply attacks, denial of service and resource consumption attacks, repudiation, and spoofing.

### 5.2 Countering Attacks

To counter the attacks, the mobile code can be written in an interpreted script or programming language. Mobile code languages can be made “type safe”. The mobile code can be signed and code capabilities can be limited. Resources can be constrained (e.g., lifetime, storage, CPU computation, etc.) and service access can be controlled (e.g., network destinations, directory segment, file system). The capabilities can be location dependent. All mobile code activities can be audited and versioned.

Users and host platforms can be issued public key certificates and can be strongly authenticated. Mobile code and messages can be conveyed securely with confidentiality, integrity, source authentication, and non-repudiation among host platforms. Replay attacks against host platforms can be detected. All platform services can be audited and versioned.

### 5.3 Securing Mobile Code

To further secure mobile code, we may prohibit malicious host platforms from further intercourse. We may allow mobile code to travel only among trusted network of platforms and to travel only one hop away from home, however that restricts autonomous activity. We may equip mobile code with tripwire objects to detect tampering or conceal the mobile code from the receiving host.

We may subject the mobile code to state appraisal as a compliment to signed code and require it to maintain path histories of the platforms visited. We may require the mobile code to convey proof of safety properties of its code and develop it in a manner that permits encrypted computation on a malicious platform.

## 6 Conclusion and Future Work

Mobile and wireless security is not much different from wired security. However, wireless has some unique difficulties, such as limited bandwidth, high latency and unstable connections.

Conventional security schemes provide a certain amount of security, but we believe the best scheme is one based on end-to-end encryption using asymmetric cryptography, where not even the network carrier control center has access to the data being transferred. To achieve this, the client machine and the information server must each perform encryption/decryption as appropriate, depending on the direction of the transmission.

This approach is independent of any security that the network provides and the cost of carrier-provided encryption may be much higher than end-to-end encryption implemented on the user device. Further more, to secure mobile code we must provide strong cryptographic mechanisms to guard against malicious hosts and to ensure that the mobile code has not been tampered with while in transient.

Related work fall into four categories: Real-time mobile operating systems, network security protocols, cryptography, mobile computing and agent systems.

The use of mobile operating systems for the Internet is relatively a new concept inspired by the proliferation of Java Virtual Machines in Web browsers. The mobile OS attempts to define a set of basic services essential to migratory, cross-platform mobile applications. The JavaOS, by Sun Microsystems, is designed to be used in network computers embracing the concept of the network as an OS [15]. However, network computers adopt the client/server model while a mobile code based OS embraces a code model that contains intermediate hosts and mobile nodes such as handheld and wireless devices.

The mobile agent paradigm was introduced several years ago. The pioneer projects in the mobile agent paradigm include Telescript and Messengers. Since the introduction of this paradigm other projects have been developed, including CyberAgent by FTP Software and Aglets by IBM. CyberAgent is a framework written in Java with primary focus on Intranet management [16]. The CyberAgents execution model assumes one agent, rather than a cooperating group of agents. IBM's Aglets defines a general-purpose mobile agent framework, but lacks essential functionality such as inter-agent communication and dynamic agent generation.

## References

1. M. Chuah: "Security-Oriented Extension to Mobile IP (SOMIP)", Internet Draft, draft-chuahlimobileip-somip-00.txt, April 24, 1997.
2. W. Diffie and M. Hellman: "New directions in cryptography", IEEE Transactions on Information Theory, IT-22(6), pp. 644-654, November 1976.
3. S. Jacobs: "Mobile IP Public Key Based Authentication", Internet Draft, March 1999.
4. B. Kaliski and M. Robshaw: "Message authentication with MD5," RSA Laboratories CryptoBytes, Vol. 1, No. 1, Spring 1995.
5. C. Kaufman, R. Perlman, and M. Speciner: "Network Security: Private Communication in a Public World", Prentice Hall, Upper Saddle River, NJ, 1995.
6. P. Kocher: "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", http://www.cryptography.com/timingattack/paper.html
7. Sufatrio and K. Lam: "Scalable Authentication Framework for Mobile-IP (SAFe-MIP)", Internet Draft, November 1999.
8. C. Perkins: "IP Mobility Support", RFC2002, October 1996.
9. R. Rivest: "The MD5 Message-Digest Algorithm", RFC1321, April 1992.
10. J. Solomon: "Mobile IP: The Internet Unplugged", Prentice Hall, Upper Saddle River, NJ, 1998.
11. G. Tsudik: "Message authentication with one-way hash functions", ACM Computer Communications Review, 22(5), pp. 29-38, 1992.
12. J. Zao, et al.: "A public-key based secure Mobile IP", Wireless Networks, Volume 5, pp. 373-390, October 1999.
13. The WAP Forum: http://www.wapforum.org
14. Wireless Transport Layer Security Specification: WAP-261-WTLS-20010406-a, April 6, 2001.
15. Marc Abrams, ed.: "World Wide Web - Beyond the Basics", Prentice Hall, 1998.
16. Ray Burns, Mary Quinn: "The CyberAgent Framework", FTP Software.