A GENERAL SECURITY INFRASTRUCTURE FOR WIRELESS COMMUNICATION

W. FENG

Los Alamos National Laboratory, P.O. Box 1663, Los Alamos NM 87545, USA
E-mail: feng@lanl.gov

J. AL-MUHTADI

University of Illinois at Urbana-Champaign, 1304 W. Springfield Ave., Urbana, IL 61801, USA
E-mail: almuhtad@uiuc.edu

The convergence of mobile technology with Internet connectivity promises to revolutionize the way we access services, run applications, and perform daily activities. However, security in wireless environments still suffers from weaknesses, vulnerabilities, and exposures to cyber-threats. We briefly examine limitations in existing wireless security schemes and present a general-purpose security infrastructure that overcomes these shortcomings.

1 Introduction

Mobile computing will revolutionize the way we access services and use computing. The major advances in mobile devices and wireless networking have converged to enhance global interconnectivity and to enable the idea of ubiquitous computing where mobile users can access services, run programs, utilize resources, and harvest computing power anytime and anywhere. This new generation of ubiquitous and mobile computing enables the delivery of integrated services and multimedia-enabled applications that are no longer bound by time or location barriers. These types of capabilities are among the major driving forces for the 3rd Generation (3G) of wireless communication and beyond [15].

Although wireless communication provides greater flexibility and mobility, such features often come at the expense of security. This is because wireless communication relies on an open and public transmission media over which eaves-dropping, unauthorized access, user tracking, and other security threats can be carried out more effectively in comparison with wired networks. Further, since wireless devices carry significantly less processing power and memory than their wired counterparts, they are often considered easier targets to launch security attacks against.

To fully realize the potential of 3G mobile devices and beyond, we must address the perceived lack of security that exists in current wireless technologies. In

this paper, we explore the problems and shortcomings in existing wireless security technologies and propose a general-purpose infrastructure to overcome the limitations, and thus, secure wireless communication more effectively. We refer to our infrastructure as *IRIS: Inter-Realm Infrastructure for Security.*

2 Existing Technologies

The Wireless Application Protocol (WAP) [17] is an industry-initiated standard for delivering information and services to wireless devices such as mobile phones and handheld devices. WAP defines a set of protocols at the network, transport, session, and application layers. Among the defined protocols is the Wireless Transport Layer Security (WTLS), which provides critical security services for mobile devices. WTLS is based on TLS [5] but is optimized for wireless networks. To provide Internet connectivity, a WAP gateway operates between the wired and wireless networks. The WAP gateway translates between the WAP protocol, which is optimized for low bandwidth and restricted resources, and the TCP/IP protocol.

iMode [2][8] is a wireless Internet service that uses proprietary protocols owned by NTT DoCoMo of Japan. iMode has successfully implemented and deployed TLS (Transport Layer Security, also known as SSLv4) over wireless, enabling iMode compliant devices to communicate securely with Internet servers.

The 3rd generation mobile system UMTS (Universal Mobile Telecommunication System) [15] focuses on designing an "all-IP" architecture for voice communication and Internet connectivity. The specification of UMTS suggests the deployment of mobile IP as the communication protocol for this architecture [16]. As a result, IP security (IPsec) [9] will be used to secure network communication.

2.1 Lack of Reconfiguration

As different generations of mobile communication come and go, mobile devices with significantly different capabilities will continue to co-exist. Imposing a fixed security standard (or protocol) for securing wireless communication leads to systems that are inflexible. Additionally, such systems become vulnerable or even unusable whenever a security flaw is discovered in the protocol or in one of the employed cryptographic algorithms, e.g., GSM's A5/1 encryption algorithm [3] and the 128-bit version of WEP, which is employed in wireless LANs [14]. Lastly, many existing approaches offer security as an all-or-nothing option.

For the above reasons, wireless communication needs security services that can be dynamically reconfigured allowing them to adapt to different scenarios, security requirements, and computing resources. Therefore, instead of having a single security protocol that is hardwired into the device, support for multiple security protocols yields greater flexibility and compatibility. Our approach provides a thin

middleware layer that allows applications and services to invoke the needed security services in a general fashion.

2.2 *Security Gaps*

Ashley et al. [2] note that security gaps appear when a secure session is terminated prematurely. Such terminations often occur in wireless communication due to the multi-mode nature of the communication link between the mobile device and its final destination, resulting in security gaps that can expose sensitive data. For example, WAP-enabled devices access Internet services through a WAP gateway. To enable a "secure" connection, a session is established between the wireless device and the WAP gateway using the WTLS protocol, as shown in Figure 1. An SSL session is then established between the WAP gateway and the application server providing the requested service. Because of the premature termination and the re-establishment of the secure session, data resides in an insecure state on the WAP gateway.

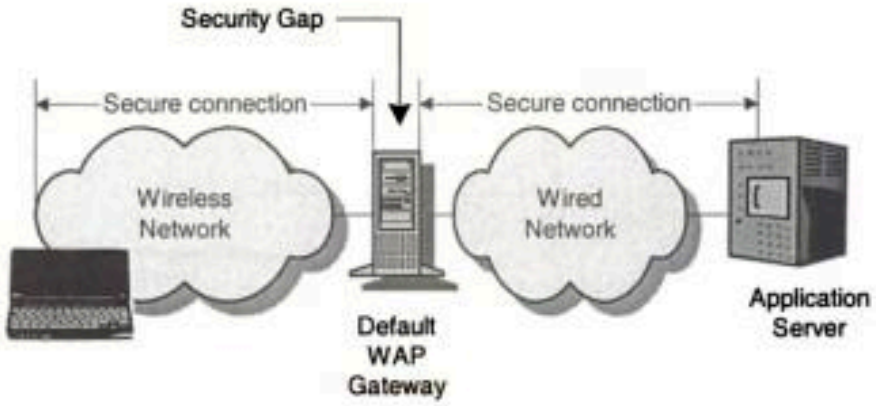

Figure 1. The WAP Security Gap.

Because iMode deploys TLS, it is technically possible to establish an end-to-end secure connection between communicating parties. However, the specifications of iMode are proprietary, and hence, are not public available. Nevertheless, there is speculation that iMode also introduces security gaps by establishing two separate TLS sessions, the first between the mobile device and the iMode server and the second between the iMode server and the application server [2].

3G's all-IP network for mobile devices calls for the use of IPsec to secure communications. However, 3G's architecture only uses IPsec between different network realms and is not truly end-to-end, as shown in Figure 2. Because IPsec generally operates only between the realm boundaries for institution A and institution B, security gaps *within* each of these realms are introduced.

The Internet (with both wired and wireless portions) is divided into different areas, each having diverse properties and characteristics such as different link-layer technologies, media types, security requirements, bandwidth capacities, and so on. We refer to areas with similar network characteristics as *realms* [7]. Special devices at the realm boundaries exist to handle the diversities. These devices transparently fix packet flows between endpoints, handle data transition between realms, and provide important additional functionality. The added functionality includes (but is not limited to) mobility support, address translation, packet filtering, and data compression. Such special devices are referred to as middleboxes. Examples of middleboxes include WAP gateways, iMode servers, home/foreign agents, firewalls, proxies, and network address translators (NATs). Unfortunately, all existing security protocols do *not* take into account the existence of middleboxes.

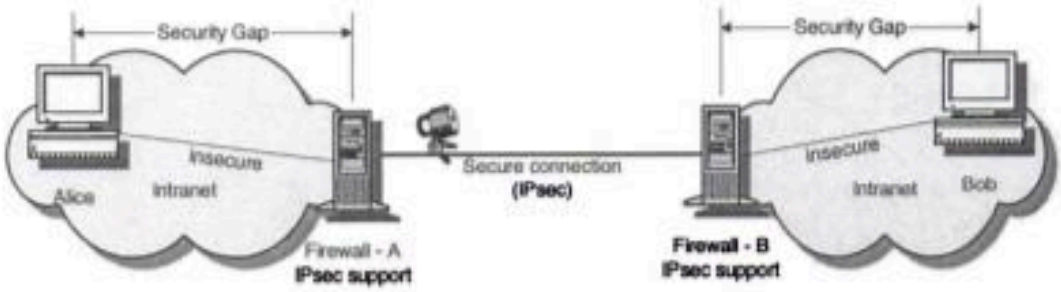

Figure 2. Security Gaps in Current IPsec Implementations.

2.3 Communication-Protocol Dependence

Additionally, many existing security protocols depend on a particular communication protocol. The protocol-dependency limits their portability to other networking infrastructures. For example, IPsec inherently depends on IP whereas many wireless environments do not use IP for communication, e.g., WAP and distributed sensor networks [6].

3 IRIS Architecture

We propose a security infrastructure that incorporates greater flexibility and adaptability, in particular, the ability to capture the dynamics and agility of mobile environments. When it comes to security, one size does *not* fit all. Hence, our proposed security architecture must be able to adapt to environments with extreme conditions and scarce resources, e.g., distributed sensor networks, and to evolve by providing additional functionality when more resources become available. The architecture should also support multiple security mechanisms and negotiate security requirements. We base the security services in our IRIS system on available and proven

technologies, thereby granting us flexibility and secure interoperability with existing systems. We, however, plan to enhance these technologies with a component-based design of a security infrastructure that includes the discovery of middleboxes and the negotiation of security requirements allowing secure communication over different realms without the need to introduce security gaps. Figure 3 illustrates the system's architecture, which is described below.

3.1 The Lightweight Core

IRIS provides a thin and lightweight middleware layer that exports three sets of APIs that are available for applications' use: GSS-API (Generic Security Services API [11][12], CM-API (Component Management API), and SI-API (Session Initiation API). These APIs make up the lightweight "core" of the system. This core can be pre-loaded into all devices. We designed the core to be small enough to fit into most existing mobile devices. Additional functionality can then be loaded on demand and plugged into the core. More information about the memory footprint required by our IRIS prototype implementation will be presented later in this paper.

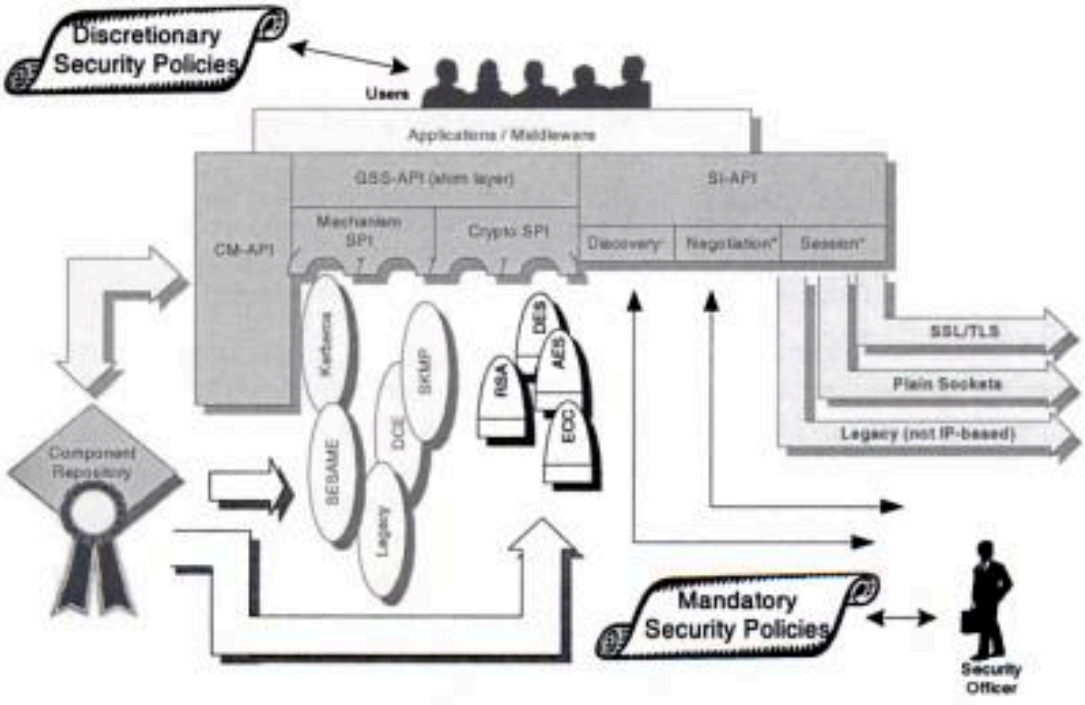

Figure 3. System Architecture of IRIS.

3.2 *GSS-API: Generic Security Services API*

The GSS-API in IRIS exports a uniform, generic interface for providing security services from an underlying security mechanism. The underlying security mechanism can be any one of the following commonly used, security mechanisms: SPKM (Simple Public Key Management Protocol), Tiny SESAME [1] (a lightweight version of a Kerberos extension), or legacy protocols. The GSS-API provides an interface that is independent from both the underlying security mechanism and the programming environment. This abstraction enables security mechanisms to be removed, added, and updated without affecting the applications. Moreover, GSS-API provides security services that are independent from the communication protocol suite being used. The GSS-API can be extended to support an arbitrary list of underlying mechanisms. Further, the extension can take place dynamically such that the necessary security protocols and cryptographic functions are loaded on demand. Hence, only the functionality needed at a certain time is loaded. This method facilitates the incorporation of new security technologies and bug fixes as they become available.

To support several security mechanisms at once and to be able to query available mechanisms and load/unload them as necessary, we follow the recommendation of [13] where the GSS implementation consists of two parts: (1) the GSS-API shim layer, which does *not* provide any security but exports a standardized security interface and (2) underlying security mechanisms and supporting cryptographic profiles, which are added to implement the actual security services. SPIs (Service Provider Interfaces) allow the GSS to locate and query the different security mechanisms and cryptographic functions.

3.3 *Loadable Components and the Component Repository*

IRIS implements security mechanisms and cryptographic functions as add-on components, components that are not part of the pre-loaded lightweight core. Such a component-based architecture enhances the adaptability of IRIS, allowing it to unload unnecessary functionality to compensate for shortages in resources and to reload that functionality when resources become more available.

IRIS stores the components that implement additional functionality in a *component repository*. To prevent the loading of malicious components, a trusted certificate authority certifies the repository and digitally signs its components to prove their authenticity, and hence, protect them against tampering. Mobile devices can then connect to the repositories to download new functionality or to update their existing security services.

3.4 CM-API: Component Management API

Because the various functionalities (including the security mechanisms and cryptographic libraries) are represented as components, IRIS provides a Component Management API (CM-API) to facilitate the management, loading, unloading, and reconfiguration of components as well as the validation of component repositories. The CM-API exposes an interface for security-aware applications to manage available components.

The CM-API is also utilized by the GSS-API to load and unload functionality on demand. End users and applications then use the CM-API to express discretionary security policies by tagging the loadable components as required, preferred, allowed, or prohibited. This provides end users and applications with flexibility in defining additional security requirements.

3.5 SI-API: Session Initiation API

The Session Initiation API (SI-API) in IRIS provides a session-layer library for applications. It can be used for the discovery of middleboxes and for negotiating security requirements between these middleboxes. We discuss this discovery process in the next section. This communication session can be based on plain sockets (in conjunction with GSS-API services to secure transmitted data), or it can use SSL (or SSL-based) protocols for interoperability with existing systems. The implementation of such protocols can be loaded dynamically as well.

3.6 Discovery of Middleboxes

IRIS discovers middleboxes by employing a session-signaling protocol, as suggested in [7]. Requirements for the discovery of middleboxes are addressed in [10]. We give an overview of how such a protocol works in IRIS. If two endpoints (*A* and B) decide to communicate securely, the sender *A* initiates the session-layer signaling protocol by sending a special "discover" request along the path to the end destination (see Figure 4). In this scenario, we assume that endpoint *A* supports security mechanism *X*. Middleboxes along the path reply to the "discover" request to indicate their presence and their security requirements, if any. In the example depicted in Figure 4, the first middlebox along the path is a wireless gateway.

The gateway supports mechanisms *Y* and *Z* and requires all outbound traffic to use one of these mechanisms. The gateway responds to the "discover" request announcing its existence and the supported security mechanisms (*Y* and *Z*). Because no common mechanisms exist between endpoint *A* and the gateway, only partial negotiations take place at this time. Proceeding with the discovery process, eventually endpoint *B* will respond and communicate its presence and supported mechanisms (*X* and *Y*).

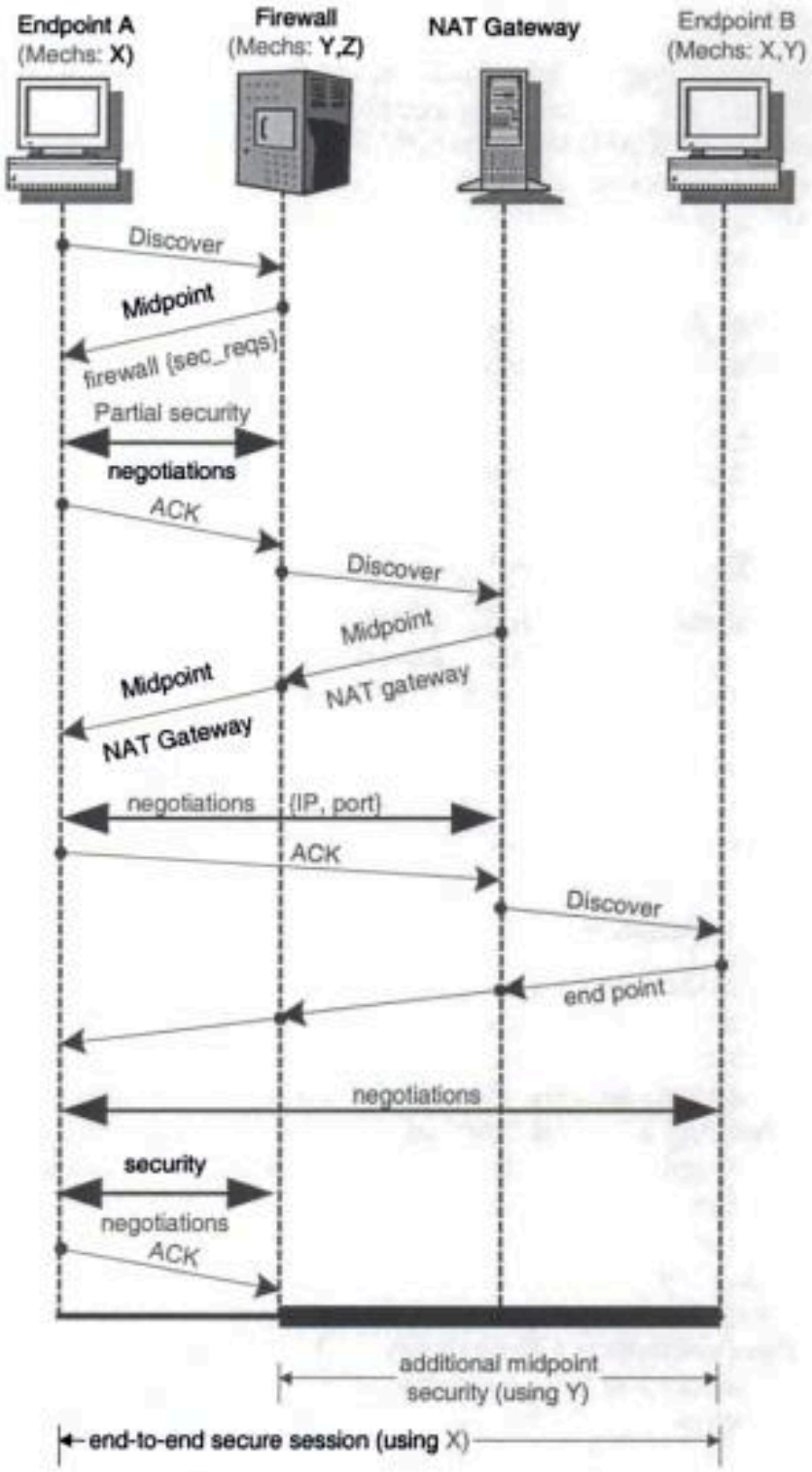

Figure 4. Discovery of Middleboxes.

Now, endpoint *A* can negotiate the establishment of two layers of security: (1) an end-to-end security session established at endpoint *A*, using mechanism *X*, with the intention to terminate only at endpoint *B* and (2) an additional midpoint security session that is established from the wireless gateway to endpoint *B*, using mechanism *Y*, to meet the additional security requirements imposed by the gateway on outbound traffic. Note that no security gaps are introduced in this scenario.

Similarly, networks may be protected by firewalls or proxies that require certain forms of authentication before network traffic can pass through them. The discovery protocol can locate such firewalls and negotiate a sufficient GSS-API-based authentication mechanism. Figure 4 depicts a scenario in which a NAT gateway exists somewhere along the path. Likewise, the NAT announces its pres-ence, and negotiations kick-in during which time the destination IP address and port are communicated, allowing the NAT gateway to deliver the data to the original destination without introducing security gaps (e.g., through tunneling). Alternatively, Realm-Specific IP (RSIP) [4] and similar protocols can be employed in place of NAT. RSIP is a proposed standard that allows a system to allocate a global address from a local gateway. The system then uses that global IP for outside communication but tunnels the traffic to the local gateway. In the reverse direction, the gateway maintains a table of allocated addresses and tunnels incoming traffic to the appropriate local system.

4 System Implementation

Our current IRIS prototype is implemented in pure Java. We use Java due to its cross-platform compatibility and the fact that Java is rapidly gaining ground in the mobile world. An increasing number of mobile phones and PDAs already support Java 2 Micro Edition (J2ME), which is a Java 2 platform that targets wireless and embedded devices.

For this paper, we chose to implement Tiny SESAME [1] as a loadable, underlying security mechanism. Tiny SESAME itself is component-based and written in Java. Other security mechanisms can be added easily. Loadable security mechanisms do not need to be ported to Java to be added to IRIS, but they do have to be wrapped by a "component wrapper" which facilitates the interfacing between the IRIS core and the loadable components. New cryptographic algorithms are implemented as Java cryptographic service providers. Table 1 shows the memory footprints consumed by the IRIS prototype.

In our current prototype, the component repository is implemented as a standalone server that resides on a wired network. Components can be queried and downloaded remotely. Components are digitally signed by a trusted third party as a protection against malicious code. Since component repositories can be distributed on a global scale, we use Jini technology from Sun Microsystems, which provides an

infrastructure for discovering and delivering services in a distributed computing environment.

Table 1. IRIS Memory Footprint.

IRIS Core	28 KB
Tiny SESAME (core, no security services loaded)	31 KB
Tiny SESAME (exportable-level security, password-based authentication)	80 KB
Tiny SESAME (full-crypto library, X.509 certificate authentication)	271 KB

5 Conclusion

The full potential of mobile computing and wireless communication can only be realized once an adequate, inter-realm security infrastructure is devised and deployed. We discussed the requirements of such an infrastructure and designed and implemented an architecture that provides comprehensive security services in a component-based fashion, providing greater adaptability, customizability and backward-compatibility.

References

1. Al-Muhtadi J., Mickunas D., and Campbell R., A Lightweight Reconfigurable Security Mechanism for 3G Mobile Devices, *Int'l Conf. on 3rd Generation Wireless and Beyond* (2001).
2. Ashley P., Hinton, H. and Vandenwauver M., Wired versus Wireless Security: The Internet, WAP, and iMode for E-Commerce, *17th Annual Computer Security Applications Conference* (2001).
3. Biryukov A. and Shamir A., Real-Time Cryptanalysis of the Alleged A5/1 on a PC, *Preliminary Draft* (1999).
4. Borella M. and Lo J., Realm Specific IP: Framework, *Internet Draft <draft-ietf-nat-rsip-framework-05.txt>* (2000). Work in progress.
5. Dierks T. and Allen C., The TLS Protocol, *RFC 2246* (1999).
6. Elson J. and Estrin D., Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks, *21st Int'l Conf. on Distributed Computing Systems* (2001).
7. Fisk, M. and Feng, W., Interactions of Realm Boundaries and End-to-End Network Applications, *Los Alamos Unclassified Report 00-3631* (2000).

8. iMode, All About iMode Index, *NTT DoCoMo, http://www.nttdocomo.com /i/index.html.*
9. Kent S. and Atkinson R., Security Architecture for the Internet Protocol, *RFC 2401* (1998).
10. Lear E., Requirements for Discovering Middleboxes, *Internet Draft <draft-lear-middlebox-discovery-requirements-00.txt>* (2001). Work in progress.
11. Linn J., Generic Security Service Application Program Interface, *RFC 1508* (1993).
12. Linn J., Generic Security Service Application Program Interface Version 2, *RFC 2078* (1997).
13. Smith M., A Service Provider API for GSS Mechanisms in Java, *Internet Draft <draft-ietf-cat-gssv2-javabind-spi-02.txt>* (1999). Work in progress.
14. Stubblefield A., Loannidis J., and Rubin A., Using the Fluhrer, Mantin, and Shamir Attack to Break WEP, *AT&T Labs Technical Report TD-4ZCPZZ* (2001).
15. UMTS, 3rd Generation Partnership Project (3GPP, UMTS Specification), http://www.3gpp.org/.
16. UMTS, TSG SA, Architecture Working Group, ftp://ftp.3gpp.org/TSG_AS/WG2_Arch/.
17. WAP, The WAP Forum, http://www.wapforum.org.