International Journal of Academic Research and Development
ISSN: 2455-4197
Impact Factor: RJIF 5.22
www.academicsjournal.com
Volume 2; Issue 5; September 2017; Page No. 907-911

A study on wireless service and its security

[*1] Santanu Sikdar, [2] Dr. RP Singh
[1] Department of UTD, Sri Satya Sai University of Technology and Medical Sciences, Sehore, Madhya Pradesh, India
[2] Department of Computer Science & Engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore, Madhya Pradesh, India

Abstract
Internet-enabled wireless devices continue to proliferate and are expected to surpass traditional Internet clients in the near future. This has opened up exciting new opportunities in the mobile e-commerce market. However, data security and privacy remain major concerns in the current generation of "wireless Web" offerings. All such offerings today use a security architecture that lacks end-to-end security.
This unfortunate choice is driven by perceived inadequacies of standard Internet security protocols like SSL on less capable CPUs and low-bandwidth wireless links. This article presents our experiences in implementing and using standard security mechanisms and protocols on small wireless devices.
We have created new classes for the Java 2 Micro-Edition platform that offer fundamental cryptographic operations such as message digests and ciphers as well as higher level security protocols like SSL. Our results show that SSL is a practical solution for ensuring end-to-end security of wireless Internet transactions even within today's technological constraints.

Keywords: security, wireless, internet

Introduction
In the past few years, there has been explosive growth in the popularity and availability of small handheld devices (mobile phones, PDAs, pagers) that can wirelessly connect to the Internet. These devices are predicted to soon outnumber traditional Internet hosts like PCs and workstations. With their convenient form factor and falling prices, these devices hold the promise of ubiquitous (anytime, anywhere) access to a wide array of interesting services. However, these battery driven devices are characterized by limited storage (volatile and nonvolatile memory), minimal computational capability, and screen sizes that vary from small to very small. These limitations make the task of creating secure, useful applications for these devices especially challenging.
It is easy to imagine a world in which people rely on connected handheld devices not only to store their personal data, and check news and weather reports, but also for more security-sensitive applications like online banking, stock trading, and shopping — all while being mobile. Such transactions invariably require the exchange of private information like passwords, PINs, and credit card numbers, and ensuring their secure transport through the network becomes an important concern [1].
On the wired Internet, Secure Sockets Layer (SSL) [1] is the most widely used security protocol.[2] Between its conception at Netscape in the mid-'90s and standardization within the Internet Engineering Task Force (IETF) in the late '90s, the protocol and its implementations have been subjected to careful scrutiny by some of the world's foremost security experts [2].
No wonder, then, that SSL (in the form of HTTPS which is simply HTTP over SSL) is trusted to secure transactions for sensitive applications ranging from Web banking to securities trading to ecommerce. One could easily argue that without SSL, there would be no e-commerce on the Web today. Almost all Web servers on the Internet support some version of SSL. Unfortunately, none of the popular wide-area wireless data services today offer this protocol on a handheld device.
Driven by perceived inadequacies of SSL in a resource-constrained environment, architects of both WAP and Palm.net chose a different (and incompatible) security protocol (e.g., WTLS [3] for WAP) for their mobile clients and inserted a proxy/gateway in their architecture to perform protocol conversions. A WAP gateway, for instance, decrypts encrypted data sent by a WAP phone using WTLS and re encrypts it using SSL before forwarding it to the eventual destination server. The reverse process is used for traffic flowing in the opposite direction. Such a proxy-based architecture has some serious drawbacks.
The proxy is not only a potential performance bottleneck, but also represents a "man-in-the-middle" privy to all "secure" communications. This lack of end-to-end security is a serious deterrent for any organization thinking of extending a security-sensitive Internet-based service to wireless users. Banks and brokerage houses are uncomfortable with the notion that the security of their customers' wireless transactions depends on the integrity of the proxy under the control of an untrusted third party.
As wireless data services evolve, their architects are faced with two choices that can profoundly impact the future of the wireless Internet. They can adopt (if necessary, adapt) standard Internet protocols, or create an entirely different set

of standards applicable only in the wireless world. The former choice would seamlessly extend the Internet to future mobile devices, the latter could severely stunt its expansion.

The Wireless Application Protocol (WAP) Forum subscribed to the "wireless is different" philosophy for its WAP 1.0 specification which defines an entire suite of protocols that parallel standard TCP/IP and web protocols, but are incompatible with them [5]. In contrast, others like the IETF's PILC working group have put forth proposals to reuse existing protocols and standards in ways that accommodate the special characteristics of wireless networks without destroying compatibility.

Proxy-Based Architecture

Due to protocol incompatibilities, a WAP device cannot communicate directly with the large installed base of Internet hosts. Instead, all communication must go through a gateway or proxy that performs protocol (and possibly content) translation. In typical deployments of WAP, this proxy is owned and maintained by a wireless service provider, who preprograms the proxy in all of its customers' phones. This allows the service provider to control what parts of the Internet are accessible to its customers, thereby creating a "walled garden" [6].

These architectural choices raise a number of concerns:

- Scalability: Since a proxy must process data packets to and from a large number of mobile devices, it represents a potential performance bottleneck (besides being a single point of failure). The insertion of a proxy also precludes end-to-end flow control. Since the wired side of a proxy has greater bandwidth than its wireless side, the proxy needs to maintain large data buffers for each active data flow.
- Legal: A recent French court ruling, prohibiting France Telecom from dictating which gateways its customers could use, is an indication that such an approach may have legal and anti-trust implications [6].
- Security: The most glaring drawback of a proxied architecture is the lack of end-to-end security. In the process of decrypting and re encrypting traffic, the proxy gets to see all communication in the clear.[3] This "WAP gap" problem is depicted in Fig. 1; even though the wired and wireless "hops" are encrypted, the proxy is privy to all information exchanged. Sometimes the situation is even worse, since weak encryption (or none at all) is used on the wireless side, giving mobile users a false sense of security. This problem is not unique to WAP.

The Palm.net security architecture uses a proprietary protocol between the wireless device and the Palm.net proxy (owned and operated by Palm). SSL is used only between the proxy and the eventual destination. Thus, when a Palm VII user accesses an HTTPS URL to establish a secure connection with a Webserver, the connection that is set up isn't truly secure. This is unacceptable to security savvy organizations; for example, Sun's corporate firewall explicitly disallows connection attempts from the Palm.net proxy.

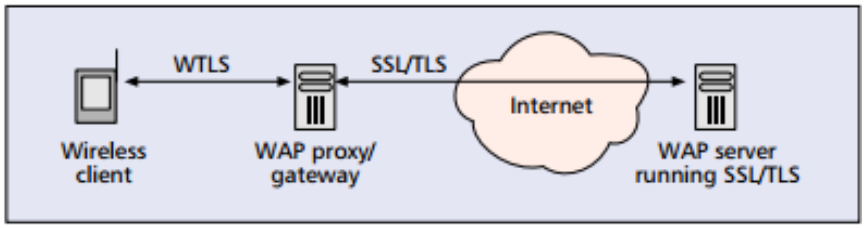

Fig 1: Proxy-based architecture

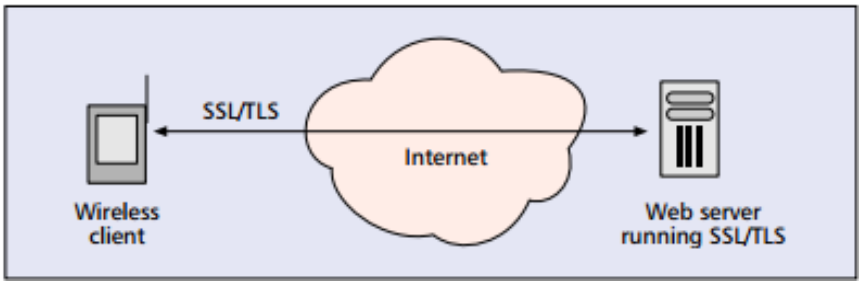

Fig 2: End-to-end architecture

End-To-End Architecture

In contrast, the use of SSL between desktop PCs/workstations and Internet servers offers end-to-end security (Fig. 2). This holds true even when an HTTPS proxy is used to traverse firewalls. Unlike the WAP or Palm.net proxy, an HTTPS proxy does not perform decryption/re encryption of data. Rather, it acts as a simple TCP relay shuttling encrypted bytes from one side to the other without modification. The expression "old is gold" is especially apt when considering security protocols.

Very often, it takes years of widespread public review and multiple iterations [7] to discover and correct subtle but fatal errors in the design and/or implementation of a security protocol. After more than five years of public scrutiny and deployment experience, 4 SSL is the most widely trusted security protocol for all sorts of web-based transactions. The addition of SSL capabilities to mobile devices would bring the same level of security to the wireless world.

Secure Sockets Layer

SSL provides encryption, source authentication, and integrity protection of application data over insecure public networks. The protocol requires a reliable bidirectional byte stream service. Typically, this service is provided by TCP, which guarantees that there is no duplication, loss, or reordering of bytes. As shown in Fig. 3, SSL is a layered protocol.

The Record layer sits above the underlying transport and provides bulk encryption and authentication services using symmetric key algorithms. The keys for these algorithms are established by the Handshake protocol, which uses public-key algorithms to create a master secret between the SSL client and server. This master secret is further used to derive cipher keys, initialization vectors, and message authentication code (MAC) keys for use by the Record layer. Until these keys are installed, the Record layer acts as a simple bidirectional pass through for all data.

Conceptually, the Alert and Change Cipher Spec (CCS) protocols sit within the same layer as the Handshake protocol. The former is used for notification of any protocol failures. The latter is used to signal successful completion of the handshake, and the start of bulk encryption and authentication in an SSL stream. SSL is very flexible and can accommodate a variety of algorithms for key agreement (RSA, DH, etc.), encryption (RC4, 3DES etc.), and hashing (MD5, SHA, etc.).

To guard against adverse interactions (from a security perspective) between arbitrary combinations of these algorithms, the standard specification explicitly lists combinations of these algorithms, called cipher-suites, with well-understood security properties. The Handshake protocol is the most complex part of SSL with many possible variations (Fig. 4). In the following subsection, we focus on its most popular form, which uses RSA key exchange and does not involve client-side authentication.

The SSL protocol allows both client- and server-side authentication. However, due to the unwieldy problem of maintaining client-side certificates, only the server is typically authenticated. Client authentication, in such cases, happens at the application layer above SSL, for example, through the use of passwords (one-time or otherwise) sent over an SSL-protected channel. The server's Certificate Request message as well as the client's certificate and Certificate Verify messages, shown in Fig. 4, are only needed for client-side authentication and rarely encountered in practice.

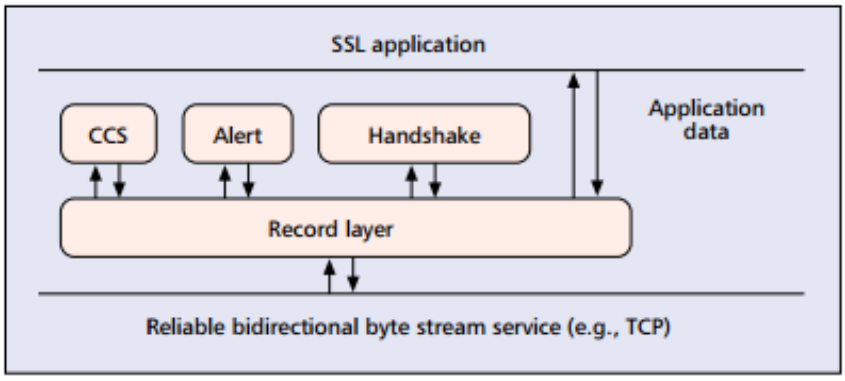

Fig 3: SSL architecture

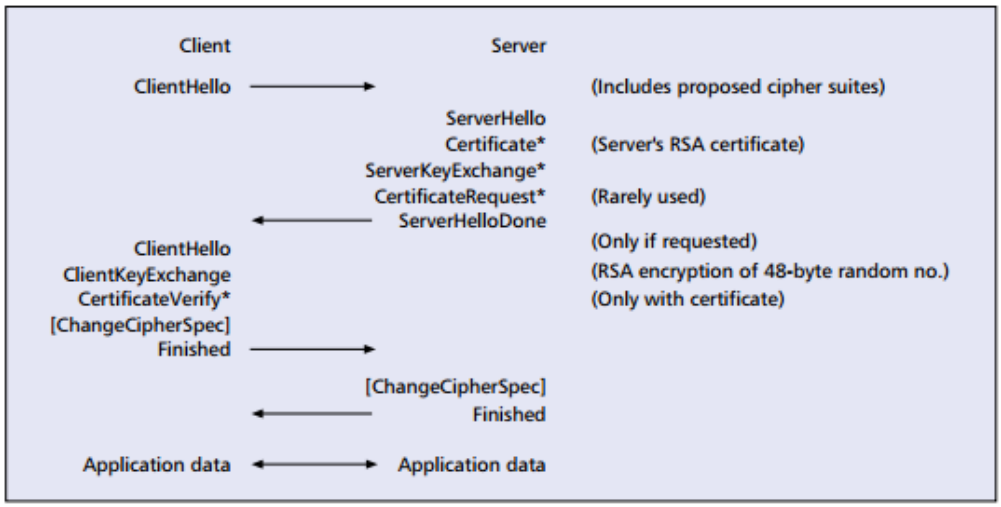

Fig 4: Full SSL handshake (comments in the third column are specific to RSA key exchange.

Full SSL Handshake

The client initiates a new SSL session by sending a Client Hello message that contains a random number (used for replay protection), a session ID (set to zero), and a set of supported cipher-suites. If the server is unwilling to support any of the proposed cipher suites, it aborts the handshake and issues a

failure notification. Otherwise, it generates a random number and a session ID and sends them in the Server Hello message along with the selected cipher suite.

The Server Hello is followed by a Certificate message containing the server's RSA public key in an X.509 certificate.[5] If this key is unsuitable for generating the Client Key Exchange message (e.g., if the key is authorized for signing but not for encryption), the server includes another RSA public key in the Server Key Exchange message and signs it with the private key corresponding to its certified public key.

The client verifies the server's public key. It then generates a 48-byte random number, called the pre-master secret, and encrypts it with the server's public key. The result is sent to the server in the Client Key Exchange message. The client also computes a master secret based on the pre-master secret, and the client and server random numbers exchanged previously. The master key is processed further to derive the symmetric keys used for bulk encryption and authentication.

These keys are installed in the record layer and a Change Cipher Spec message is sent to signal the end of in-the-clear communication. The Finished message is the first one from the client side to be secured by the negotiated cipher suite. It ensures that any tampering of prior handshake messages, sent in the clear, can be detected.

Upon receiving the Client Key Exchange message, the server decrypts the pre-master secret and follows the same steps as the client to derive the master secret and symmetric keys for the Record Layer. After installing these keys in the Record Layer, the server is able to validate the client's Finished message. The server also sends the Change Cipher Spec and Finished messages to complete the handshake. From here on, application data can be exchanged and is protected by the encryption/ hashing algorithms negotiated in the handshake. Each direction of the traffic's flow uses distinct encryption and MAC keys.

Abbreviated SSL Hand Shake

The SSL specification also supports a feature called session reuse, which allows the client and server to reuse the master key derived in a previous session. The abbreviated handshake protocol is shown in Fig. 5. Here, the Client Hello message includes the nonzero ID of a previously negotiated session. If the server still has that session information cached and is willing to reuse the corresponding master secret, it echoes the session ID in the Server Hello message.

Otherwise, it returns a new session ID, thereby signaling the client to engage in a full handshake. The derivation of symmetric keys from the master secret and the exchange of Change Cipher Spec and Finished messages is identical to the full handshake scenario. The abbreviated handshake does not involve certificates or public key cryptographic operations so fewer (and shorter) messages are exchanged. Consequently, an abbreviated handshake is significantly faster than a full handshake.

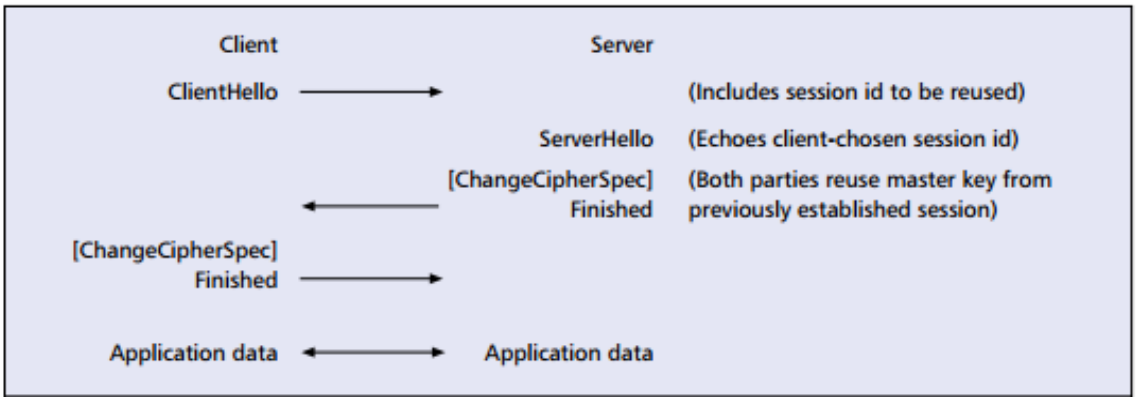

Fig 5: Abbreviated SSL handshake

Technology Trends

Since starting this project about a year ago, we've seen several examples of technology's relentless march toward smaller, faster, and more capable devices. Newer Palm PDAs like the Palm Vx and Palm IIIc use 20 MHz processors, and the Handspring Visor Platinum (another Palm OS device) features a 33 MHz processor, both considerable improvements over the earlier 16MHz CPUs. All of them offer 8 mbytes of memory. The Compaq iPaQ pocket PC, in comparison, carries a 200MHz Strong ARM processor and 16-32MB of memory. The CPU enhancements have a direct impact on the speed of SSL's cryptographic operations. Significant performance gains are also obtainable by using hardware accelerators in the form of tiny smart cards (and related devices like the I Button).

Similarly, improvements can also be seen in the speed of wireless networks. Metricom's Ricochet service now offers wireless data speeds of 128 kb/s in several U.S. cities, and 3G networks hold the promise of even faster communication in the next year or two. These improvements help reduce the network-related latency of an SSL handshake. Even with the older 32 kb/s Ricochet service, we see 15–20 percent faster handshakes compared to the CDPD network.

Powerful handhelds like the iPaQ using 802.11 for wireless connectivity have no problems whatsoever running SSL. Even smart compilation techniques, which had been available only on more capable PCs and workstations, are now available on small devices and can boost the performance of J2ME applications by as much as a factor of five. These developments should alleviate any remaining concerns about

SSL's suitability for wireless devices. They also highlight an interesting phenomenon: In the time it takes to develop and deploy new (incompatible) protocols, technology constraints can change enough to raise serious questions about their long-term relevance.

Conclusions and Future Work

Our experiments show that SSL is a viable technology even for today's mobile devices and wireless networks. By carefully selecting and implementing a subset of the protocol's many features, it is possible to ensure acceptable performance and compatibility with a large installed base of secure Web servers while maintaining a small memory footprint. Our implementation brings mainstream security mechanisms, trusted on the wired Internet, to wireless devices for the first time.

The use of standard SSL ensures end-to-end security, an important feature missing from current wireless architectures. The latest version of J2ME MIDP incorporating KSSL can be downloaded from [10]. In our ongoing effort to further enhance cryptographic performance on small devices, we plan to explore the use of smart cards as hardware accelerators and Elliptic Curve Cryptography in our implementations.

References

1. Frier A, Karlton P, Kocher P. The SSL3.0 Protocol Version 3.0"; http://home.netscape.com/eng/ssl3
2. Wagner D, Schneier B. Analysis of the SSL 3.0 Protocol," 2nd USENIX Wksp. Elect. Commerce, 1996. http://www.cs. berkeley.edu/\verb+~+daw/papers
3. WAP Forum. Wireless Transport Layer Security Specification. http://www.wapforum.org/what/technical. htm
4. Miranzadeh T. Understanding Security on the Wireless Internet, see http://www.tdap.co.uk/uk/archive/ billing/bill (phonecom_9912).html
5. Khare R. W Effect Considered Harmful," IEEE Internet Computing. 1999; 3(4):89-92.
6. Bradner S. The Problems with Closed Gardens," Network World, 2000. at http://www.nwfusion
7. Needham RM, Schroeder MD. Authentication Revisited, Operating System Review. 1990; 21(1):35-38.
8. Koblitz N. A Course in Number Theory and Cryptography 2nd ed., Springer-Verlag.
9. Boneh D, Daswani N. Experiments with Electronic Commerce on the Palm Pilot, Financial Cryptography '99, Lecture Notes in Computer Science. 1999; 1648:1-16.
10. Mobile Information Device Profile (MIDP); http://java. sun.com/products/midp