Global Information Assurance Certification Paper

Wireless Application Protocol & Security

Burak Güçer
Saglam Fikir Sok 13
Esentepe – Istanbul / Turkey
+90-212-2747375
bgucer@vis.com.tr
bgucer@superonline.com

INTRODUCTION

The Internet has proven to be easiest and most efficient way of delivering services to so called, "wired" users.

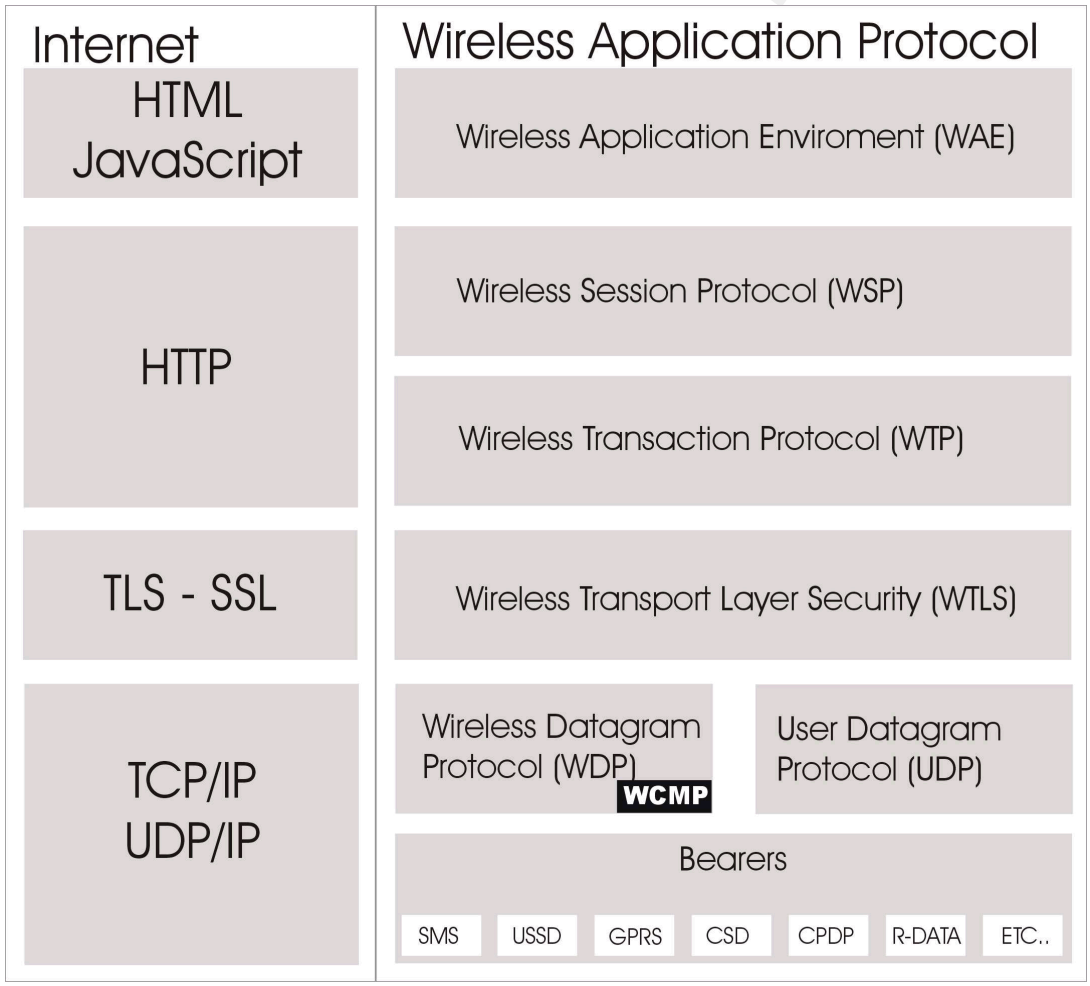
Mobile networks of today do not provide same level of desired flexibility like the "wired" networks when we consider value added services. The WAP (Wireless Application Protocol) introduces us, the concept of the Internet as wireless service platform.

In 1997, Ericsson, Motorola, Nokia and Unwired Planet founded WAP Forum, aiming to bring the convenience of the Internet into the wireless community as well.

WAP Forum has succeeded in developing a standard that scale across a wide range of wireless devices and networks. WAP is designed in a layered fashion like OSI model in order to be extensible, flexible and scalable. WAP stack basically divided into five layers:

- Application Layer: Wireless Application Environment (WAE)
- Session Layer: Wireless Session Protocol (WSP)
- Transaction Layer: Wireless Transaction Protocol (WTP)
- Security Layer: Wireless Transport Layer Security (WTLS)
- Transport Layer: Wireless Datagram Protocol (WDP)

Each layer of the WAP protocol stack specifies a well-defined interface to the layer above.



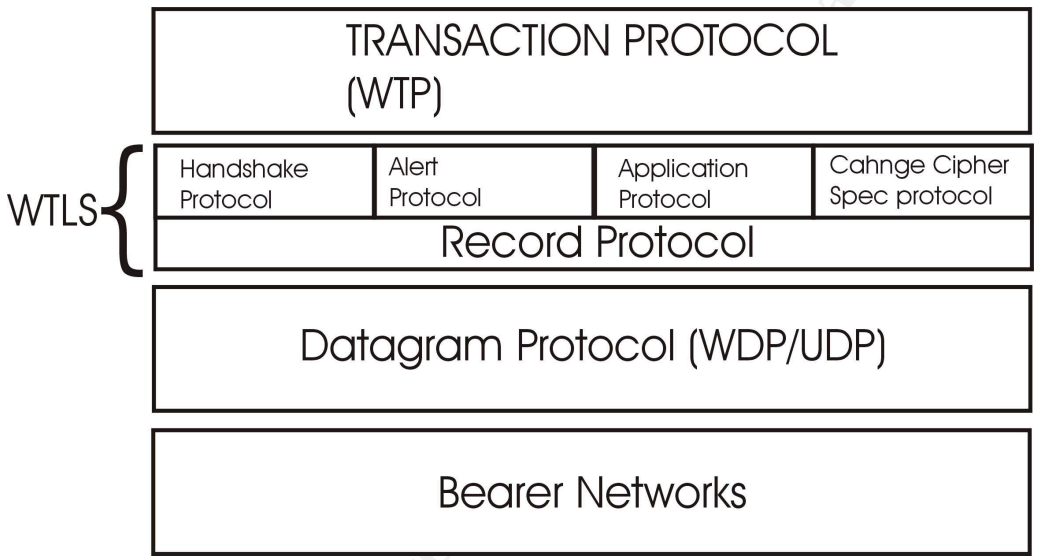
In fact, our aim is to discuss security dimension of WAP, so we will focus on WTLS. The purpose of developing WTLS is to provide transport layer security between a WAP client and the WAP Gateway/Proxy. The WTLS (Wireless Transport Layer Security) protocol is the

security layer of the WAP (Wireless Application Protocol). It is becoming the de facto standard for providing three major security issues (privacy, data integrity and authentication) for applications in cellular phones and other small wireless terminals. Millions of devices using WTLS are, expected to be fielded worldwide before the end of the year 2000.

WTLS bears a close resemblance to the SSL and TSL protocols with a number of changes, motivated by the special requirements of the WTLS, made by the WAP forum.

- Both datagram and connection oriented transport layer protocols has to be supported
- The protocol must be able to cope with long round-trip times.
- The bandwidth can be very low.
- The processing power of many mobile terminals is quite limited.
- The memory capacity of many mobile terminals is very modest.
- The restrictions on exporting and using cryptography must be considered

In other words, the authors of WTLS took TLS and tried to add datagram support, optimize the packet size, and select fast algorithms into the algorithm suite.



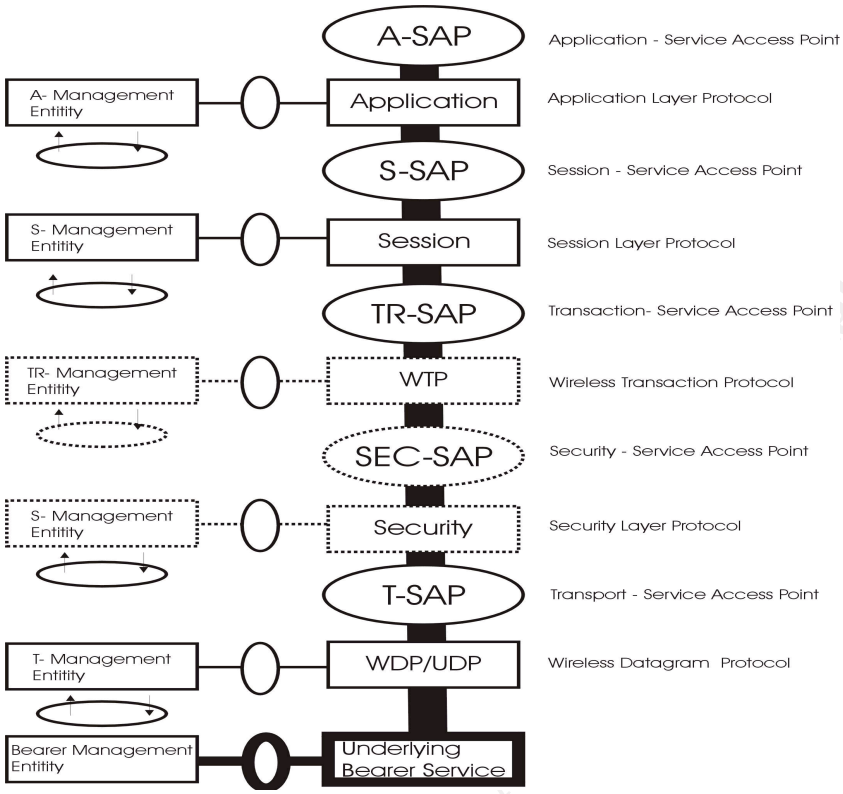
WTLS is designed to function on connection-oriented and/or datagram transport protocols and security is assumed to be an optional layer above the transport layer. The security layer preserves the transport service interfaces. The session or application management entities are assumed to provide additional support to manage secure connections (eg, initiate and terminate).

Key Features:

- Integrity through the use of Message Authentication Codes (MAC)
- Confidentiality through the use of encryption
- Authentication and no-repudiation of server and client, using digital certificates.

These features make it possible to certify that the sent data have not been manipulated by a 3rd party, that privacy is guaranteed, that an author of a message can be identified, and that both parties cannot falsely deny having sent their messages. A secure connection is setup with an establishment phase where negotiation such as parameter settings, key exchange and authentication is performed. Both parties can abort the secure connection during establishment or at any time later. WTLS is optional and can be used with both the connectionless and the connection mode WAP stack configuration. If used, it is always placed

on top of WDP (Wireless Datagram Protocol).



SECURITY

At the surface, the WTLS looks reasonably good. Regarding the research made by Markku-Juhani Saarinen, most of the text in the WTLS specification has been adopted, word for word, from TLS specification. However, many of the changes made by WAP Forum have led to security problems.

Since WTLS operates on top of datagrams, the implementation should pay special attention to prevent DoS attacks by taking into account that in some networks transport addresses may be forged relatively easy. In order to make DoS attacks harder to accomplish, it should not be possible for an attacker to break up an existing connection and/or session by sending a single message in plaintext from a forged address. In addition, the server should be careful in accepting new connection requests in plain text within an existing secure connection. Note that the server cannot just ignore them because eg, ClientHello in plaintext may be sent by a client whose connection state was lost. Special care must be taken with arbitrated and optimized handshakes in which the server switches the pending state current immediately after responding to ClientHello message. In such case, the old active state should be kept intact until the new handshake is accomplished. In other words, the server should not discard the old active state until the client responds with Finished and the handshake is completed successfully. The old active state should be restored to the current state if it is evidenced that the handshake started is invalid. For the same reason, when a client receives a plaintext ServerHello on its secure connection, it should not cause the existing secure connection broken because of unexpected message. It should keep the existing secure connection and send *unexpected_message* as a warning. When receiving plaintext HelloRequest, the client must implement a protection mechanism: ClientHello must not be sent without checking the validity of the sender address (provisioned to the client before). Client should ignore HelloRequest if received too frequently from the same address. When

using explicit sequence numbers, an implementation needs to take special precautions against an attacker inserting a record into the datastream that could cause future records from the intended sender to be discarded as duplicates. When receiving a plaintext alert, the checksum should be validated before accepting the sequence number as valid.

Although TLS was designed assuming reliable transport (such as TCP/IP), the WTLS protocol should be able to operate over an unreliable datagram transport where datagrams may be lost, duplicated, or reordered. If CBC (Cipher Block Chaining) mode is being used, which means xoring the plaintext block with the previous ciphertext block and encrypting the resulting value, this requirement makes it necessary for the *IV* (Initialization Vector) to be either contained in the packet itself (explicit IV, as in IPSec) or that the *IV* for that block is somehow derived from data already available to the recipient. WTLS uses a linear *IV* computation, even reliable transports. Please note that, previous ciphertext block is stored in an *IV*.

When a block cipher is used in CBC mode, the *IV* for encrypting each packet is computed as follows:

$$IV_s = IV_o \oplus (s \mid s \mid s \mid s)$$

where *s* is a 16-bit sequence number of the packet and IV_o is the original *IV*, derived during key generation. The plaintext blocks $P_{s,0}, P_{s,1}, \cdots$ in the packet *s* are encrypted as

$$C_{s,0} = E_k\left(IV_s \oplus P_{s,0}\right)$$
$$C_{s,i} = E_k\left(C_{s,i-1} \oplus P_{s,i}\right)_{for\ i>0}$$

Imagine an application (such as telnet), where each key press is sent as an individual packet. Lets talk about more concrete example: Vanessa enters her password into this application, and Stefanie captures all these packets. Stefanie now has blocks of type

$$C_{s,0} = E_k\left(P_{s,0} \oplus IV_0 \oplus (s \mid s \mid s \mid s)\right)$$

where $P_{s,0}$ contains an unknown letter of Vanessa's password. Note that *s* (sequence number) is known to Stefanie.

Now somehow Stefanie gets hold of Vanessa's channel. This may happen for example through an echo feature in some application. Stefanie guesses that the unknown letter in the password is *L*. Stefanie sends the following packet through Vanessa's channel:

$$P_{r,0} = L \oplus (s \mid s \mid s \mid s) \oplus (r \mid r \mid r \mid r)$$

where *r* is the sequence number of this packet. One can see that because $(r \mid r \mid r \mid r)$ cancels out in the CBC computation, a right guess $L = P_{s,0}$ leads to matching cipher texts $C_{r,0} = C_{s,0}$. In other words, this is an oracle that tells whether the guessed password letter was correct. The entire password can be discovered by brute force method, using this oracle.

While above description of the attack is highly simplified, one can see that a too easily predictable *IV* leads to situations where low-entropy secrets can be read.

This attack is similar to the attacks described by Bellovin against the IPSec protocol.

The WTLS protocols supports, among other MACs (Message Authentication Code), a 40-bit XOR MAC. The XOR MAC works by padding the message with zeros, dividing it into 5-byte blocks and xoring these blocks together. Note that this construction differs from one presented in "XOR MACs: New Methods for Authentication Using Finite Pseudorandom Functions – M.Bellare, R Guerin and P.Rogaway – Springer-Verlag, 1995"

The specification states that XOR MAC is only intended for "some devices with very limited CPU resources". The specification also tells us that that the XOR MAC "may not

provide as strong message integrity protection as SHA" when export able encryption is being used. In fact it is easy to see that the XOR MAC does not provide any message integrity protection if stream ciphers are being used, regardless of the key length.

If one inverts a bit position *n* in the cipher text, the MAC can be made to match by inverting the bit ($n \bmod 40$) in the MAC. This can be repeated arbitrary number of times. Thus, when stream ciphers are used, the XOR MAC does not provide any integrity protection.

The 40-bit DES encryption method is defined to use five bytes of keying material. Because of the parity bits contained in each byte of a DES key, there are only 5*7=35 effective key bits in five bytes. This amounts to a reduction of the keyspace by a factor of 32. We note that 56-bit DES has the correct amount of keying material (8 bytes).

The protocol clearly does not meet its requirement of reaching the best possible security level in export-weakened encryption modes.

The RSA signatures and encryption are performed according to PKCS #1, v1.5. Daniel Bleichenbacher and others have demonstrated that if the protocol includes an oracle that tells whether a given packet has a correct PKCS#1 v1.5 padding, RSA messages can be decrypted with approximately 2^{20} chosen cipher text queries. In some implementations the WTLS error messages **bad_certificate** and **decode_error** may provide such an oracle to the attacker.

It is recommended that the 2.0 versions should be used instead.

Some of the alert messages used in the protocol are sent in clear text and are not properly authenticated. Most of these messages are warnings and do not cause the session to be terminated.

Since an alert message can take up a sequence number "slot" in the protocol, an active attacker may replace an encrypted datagram with an unauthenticated plaintext alert message with the same sequence number without being detected. This leads to a truncation attack that allows arbitrary packets to be removed from the data stream. Because of this reason it is recommended, that all messages affecting the protocol state should be properly authenticated.

Under exportable keys the initial IV of each packet can be determined by an eavesdropper from the Hello messages and the sequence number alone.

The change of keys can be determined by an eavesdropper, because the **record_type** field is sent unencrypted. This field determines the type of the message; one type being the Change Cipher Spec type.

Also, the existence of encrypted error messages can be determined from the **record_type** field. The exact nature of the encrypted error messages cannot be determined.

In order to mount an exhaustive key search on symmetric cipher, one needs to have enough known or probable plaintext, so that the correct key can be recognized with trial decryption of one or more blocks.

Brute force attacks against the block ciphers in WTLS can be easily mounted, because the correct keys can be always recognized with a trial decryption of the last block of each packet.

Assume that a 64-bit block cipher is used. The last block is padded to the next full 8-byte limit by filling it with the padding length. In other words, if the last byte of $E_k^{-1}(C_i) \oplus C_{i-1}$ is *n*, the preceding *n* bytes of the plaintext must also contain this number.

If the test is passed, the key can be furthermore verified with the last block of arbitrary number of packets.

The WTLS specification includes 521- and 768-bit primes p_1 and p_2, along with generators, that are to be used in Diffie-Hellman computations. The group order of the multiplicative subgroup generated by the generator is not given.

The absence of the group order makes it impossible to check that the given public value belongs to the correct multiplicative subgroups, as in DSA and KEA.

It is known that if the group order is relatively smooth, the discrete logarithm problem becomes substantially easier to an attacker that knows the factorization of the group order.

The group order information was left out from the specifications, not because of an attempt to mount a back door into WTLS, but because the authors did not see the relevance of the group order to the security of Diffie-Hellman key exchange. The group orders of the elliptic curve groups are given. These are all prime.

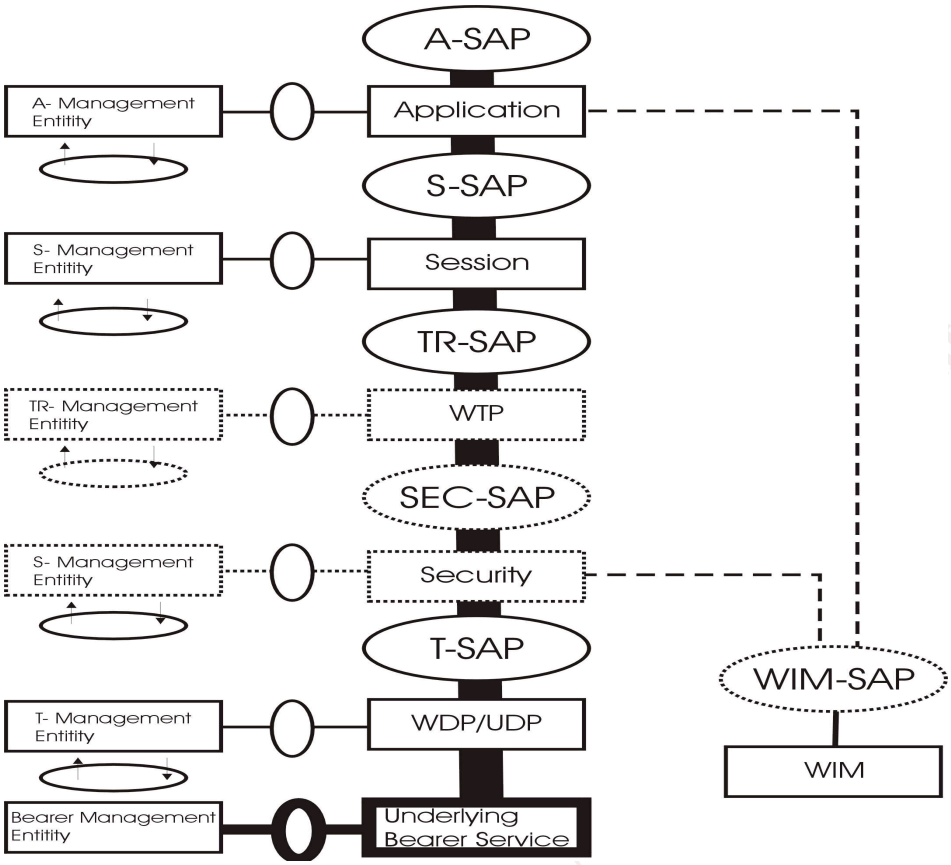
CONCLUSIONS

A number of security flaws and shortcomings in WAP WTLS protocol has been identified: a chosen plaintext data recovery attack, a datagram truncation attack, a message forgery attack, DoS and a key-search shortcut for some exportable keys. WTLS clearly needs revision.

A tamper-resistant device, which is called WIM might help for solving these security issues. It is used to enhance security of the implementation of the Security Layer and certain functions of the application layer. The WIM-SAP is defined in order to describe WIM functionality that is common to all kind of WIM implementations.

The information structure is based on PKCS15 which enables a flexible information format a cryptographic token. It uses on object model that makes it possible to access keys, certificates, authentication objects and proprietary data objects in a simple device.

The WIM functionality can be implemented on a smart card. A smart card implementation is based on ISO7816 series of standards. The WIM is defined as an independent smart card application, which makes it possible to implement it as a WIM only card or as a part of multi-application card containing other card applications, like the GSM SIM. The WIM application is designed so that it is possible to implement it with current smart card technology.



ABBREVIATIONS

API	Application Programming Interface
CA	Certification Authority
CBC	Cipher Block Chaining
DH	Diffie-Hellman
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
IV	Initialization Vector
MAC	Message Authentication Code
ME	Management Entity
OSI	Open System Interconnection
PDU	Protocol Data Unit
PRF	Pseudo-Random Function
SAP	Service Access Point
SDU	Service Data Unit
SHA-1	Secure Hash Algorithm
SMS	Short Message Service
TLS	Transport Layer Security
WAP	Wireless Application Protocol
WIM	WAP Identity Module
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol

DEFINITIONS

Abbreviated Handshake:
A creation of a new *connection state* based on an existing *secure session*.

Connection State:
The operating environment of the *record protocol*. The connection state includes all parameters that are needed for the cryptographic operations (encryption/decryption and MAC calculation/verification). Each *secure connection* has a connection state.

Datagram Transport:
A transport service that does not guarantee that the sent transport SDUs are not lost, duplicated or delivered out of order.

Handshake:
The procedure of agreeing on the protocol options to be used between a client and a server. It includes the negotiation of security parameters (eg, algorithms and key lengths), key exchange and authentication. Handshaking occurs in the beginning of each secure connection.

Full Handshake:
A creation of a new *secure session* between two peers. The full handshake includes the parameter negotiation and the exchange of public-key information between the client and server.

Optimised Handshake:
A creation of a new *secure session* between two peers. Unlike in the *full handshake*, the server looks up the client certificate from its own source without requesting it over the air from the client.

Record:
A protocol data unit (PDU) in the *record protocol* layer.

Record Protocol:
The record protocol takes messages to be transmitted, optionally compress the data, applies a MAC, encrypts and transmits the result. Received data is decrypted, verified, decompressed and than delivered to higher level clients. There are four record protocol clients described: the handshake protocol, the alert protocol, the change cipher spec protocol, and the application data protocol.

Secure Connection:
The WTLS connection that has a *connection state*. Each secure connection is identified by the transport addresses of the communicating peers.

Secure Session:

The secure session that is negotiated on handshake. The items that are negotiated (eg, session identifier, algorithms and master secret) are used fro creating *secure connection*s. Each secure session is identified by a session ID allocated by the server.

Session Resume:

A new *secure connection* can be established based on a previously negotiated secure session. So if there is an existing *secure session* it is not necessary to perform the full handshake and cryptographic calculations again. For example, a secure connection may be terminated and resumed later. Many secure connections can be established using the same secure session through the resumption feature of the WTLS *handshake protocol*.

Shared Secret Authentication

An authentication method based on a shared secret. This method works without public-key algorithms but requires that the pre-master secret is implanted or entered manually into both client and server. The shared secret is sensitive information and, therefore, a secure channel is needed for the distribution.

References

- "WAP: Wireless Transport Layer Security Specification, Version 18-Feb-2000", http://www.wapforum.org/
- "WAP Identity Module, Version 18-Feb-2000", http://www.wapforum.org/
- T. Dierks and C.Allen, "The TLS Protocol Version 1.0", RFC 2246,1999, ftp://ftp.isi.edu/in-notes/rfc2246.txt
- Saarinen, "Atacks Against the WAP WTLS Protocol",1999
- Wireless Internet Today,"WAP WhitePaper", October 1999
- E. Kristensen-WSG Chairman (Ericsson), "WAP Security", 3-Feb-99,
- A.Menezes, P.van Oorschot and S.Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996, http://www.cacr.math.uwaterloo.ca/hac