The WTLS security layer in WAP

R. Gallant

Certicom Corporation

- CACR April 2000 -

Overview

- WAP – the larger system in which WTLS lives
- the WTLS protocol itself
- other security mechanisms in WAP
- performance

WAP: Wireless Application Protocol

- A protocol for wireless applications!
- The wireless internet. Surfing with your phone, PDA, pager ...
- many big companies behind this initiative (Motorola, Nokia, Ericsson ...)
- http://www.wapforum.com
- starting to see initial deployments

WAP: Wireless Application Protocol

Surfing (information search and retrieval) is just one application:

- surfing (browsing)
- ecommerce (Amazon, ebay)
- asynchronous notification (Newsflash!, sports scores, stock prices..)
- restaurant bookings, navigation help, email, ...

In general though, WAP's architecture is analogous to internet ”browser model”

WWW 'browser' model

For us, surfing the web has the following aspects:

- browser (renders content described using HTML/JavaScript)
- browser requests pages from content servers using HTTP protocol
- optional security layer SSL (invoked for https:// pages)
- runs over TCP/IP, a reliable stream-based protocol

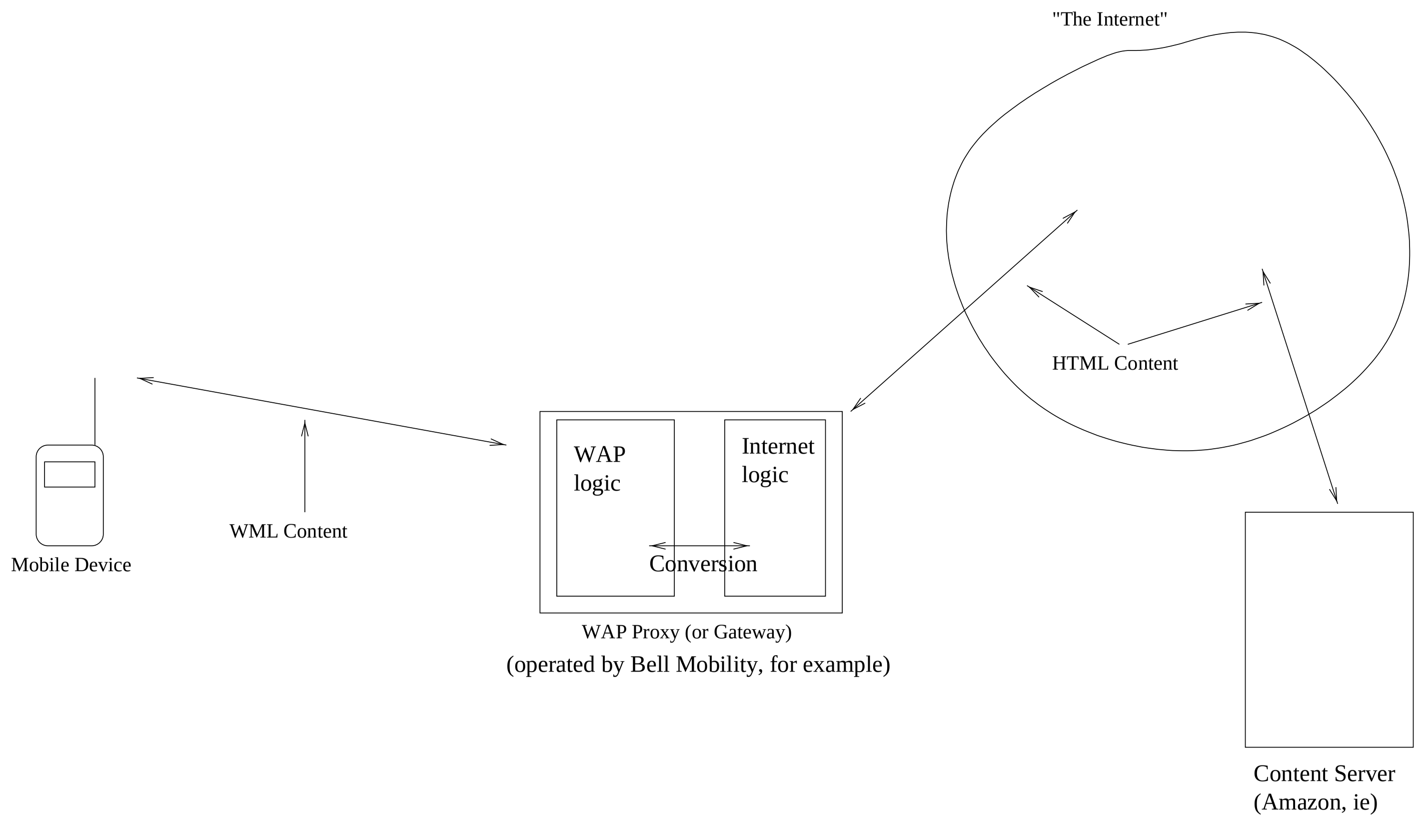

Figure 1: WAP Architecture

The WAP and WWW protocol stacks

WAP layer	**WWW/internet layer**
Browser	Microbrowser
WML/WMLScript content	HTML/JavaScript content
WSP connection	HTTP connection
WTP	(like TCP/IP)
WTLS security layer	SSL/TLS security layer
	TCP/IP
WDP datagram service	

The WTLS security layer in WAP

The Wireless Transport Layer Security (WTLS) protocol ...

- borrows heavily form the TLS (SSL) protocol
- protects packets of data (datagrams), not a stream of data.
- has a client-server framework (as is WAP in general)
- contains public key and private key components
- supports negotiation of algorithms between client and server

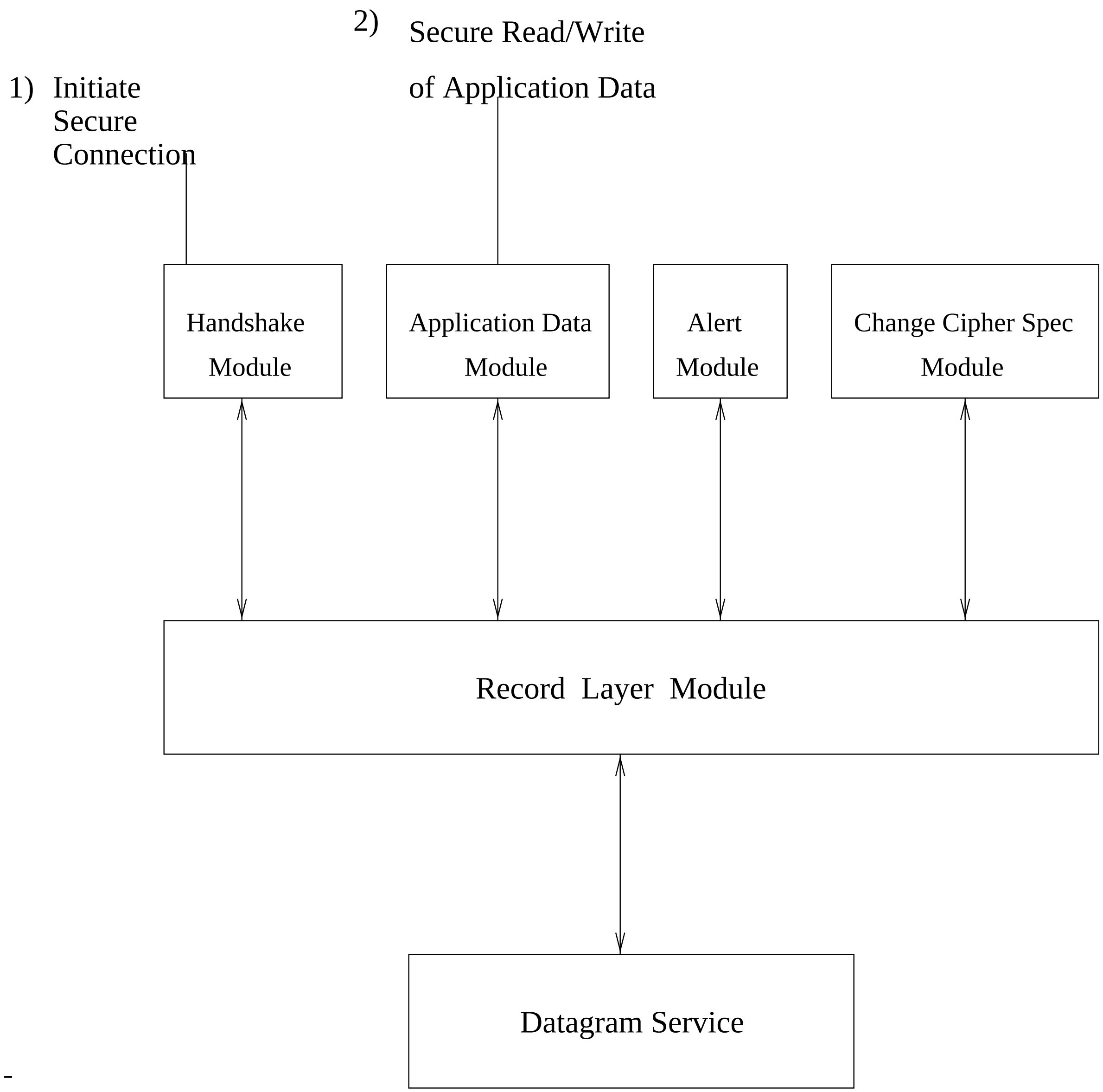

Figure 2: Architecture of the WTLS protocol

WTLS protocol: The architecture of the WTLS layer

- The 'record layer' is the packet engine for WTLS
- There are modules for the creating and parsing of
  - 'handshake' messages (public key and negotiation part of WTLS)
  - 'alert' messages (informational messages between peers)
  - 'application data' messages (the secured datagram service)
  - 'change cipher spec message' (technical, consider as part of handshake)
- Each of these modules creates module-specific packets which go into the record layer, (and so are encapsulated in a record layer packet)

WTLS protocol: the packet service (Record Layer)

The Record layer's operating state information includes

- Choice of symmetric encryption algorithm
- Choice of (symmetric) MAC algorithm
- (shared) Keys for the above algorithms
- (monotonic increasing) sequence numbers
- Optional data compression algorithm

WTLS protocol: the packet service (Record Layer)

Some of the specific algorithms available for use in the record layer:

Encryption	**Authentication**
DES-CBC	MD5-HMAC
3DES-CBC	SHA-1-HMAC
RC5-CBC	
NULL	NULL

The Record Layer uses this information to protect data on a packet by packet basis.

WTLS protocol: the packet service (Record Layer)

Some technical details concerning record layer operations...

- plaintext is first optionally compressed, then
- a sequence number and length field are prepended to the data, then
- a MAC algorithm is applied, and a tag appended to the data, then
- the data and tag is encrypted

A feature of WTLS?

WTLS is designed to protect datagrams, or packets of data.

Delivery of datagrams is not guaranteed.

Thus, WTLS cannot detect the removal of datagrams from the transmitted data. It was not designed to.

An application can trivially add this protection if desired, by just including application-level sequencing with the data.

Seeding the Record Layer

Earlier, we saw that the state information for the record layer consisted of the choice of symmetric algorithms and the associated keys.

The initial state of the Record layer is NULL: no encryption and no authentication algorithms.

The job of the *Handshake module* is

- to negotiate record layer algorithms with a peer, and
- (to use public key cryptography) to negotiate shared keys needed for these algorithms.

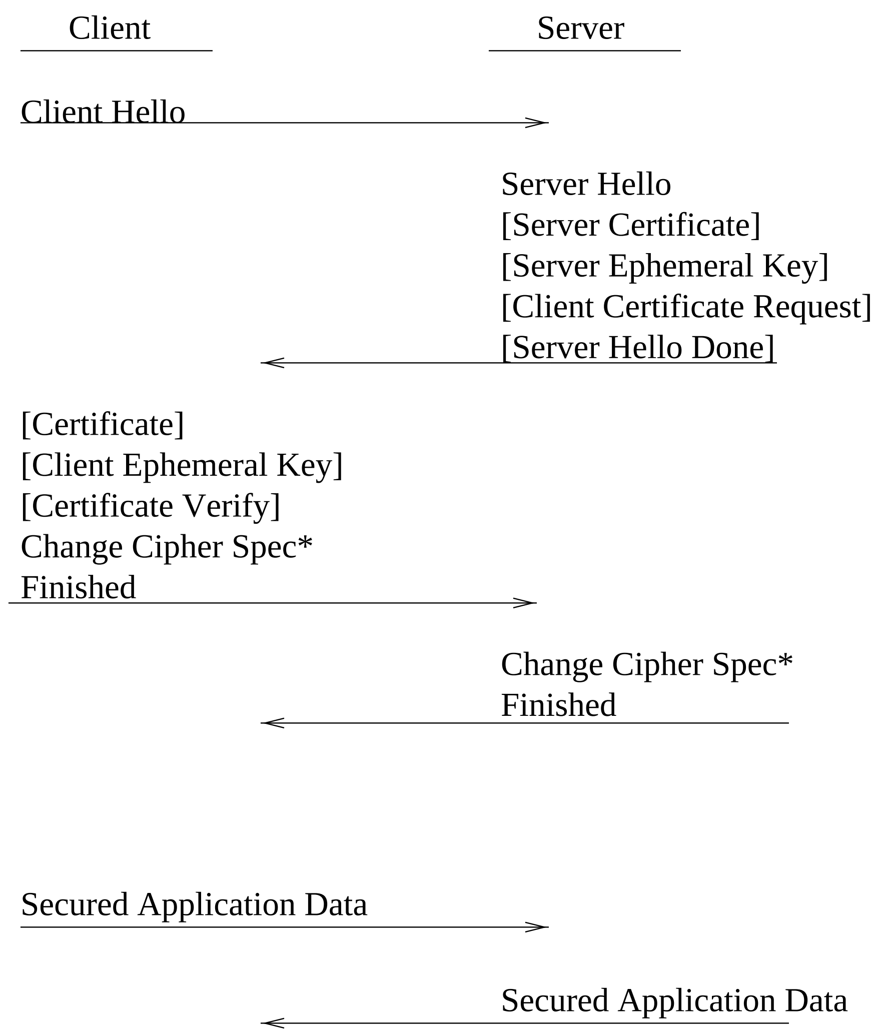

Figure 3: The WTLS (full) handshake protocol

WTLS protocol: the WTLS handshake

- In 'full' handshake, client sends negotiation options and nonce challenge to server (ClientHello)
- server selects some options and sends public key exchange data (either a certificate or an ephemeral public key)
- client responds with either it's certificate or ephemeral public key. The Change Cipher spec message indicates that all subsequent data sent (by the client, in this case) will be protected using the negotiated algorithms. The Finished message is a MAC of the handshake messages (so far), using a key derived from this shared secret.
- server responds with it's own Change Cipher Spec and Finished messages.

Public Key options in the WTLS Handshake

The following Public key algorithms are supported in the full handshake

- anonymous DH key agreement (512 and 768 bit primes)
- anonymous, uni, or bi-laterally authenticated RSA key agreement
- anonymous, uni, or bi-laterally authenticated ECDH key agreement

The WTLS standard defines groups for use in the Diffie-Hellman exchanges. The Handhake protocol also has a mechanism for negotiating new group parameters.

WTLS Handshake protocol: Technical Notes

The Handshake messages are encapsulated in Record Layer packets. If the underlying transport service is unreliable, the Handshake logic must implement retry mechanisms.

A Successful receipt of the ChangeCipherSpec message by the client results in a 'secure session'. The negotiated encryption and MAC services of the record layer are now applied to each packet sent for this session. This lasts until either Server or Client explicitly shuts down the secure connection.

For Discrete log based cryptosystems, (Elliptic curve and F_p^*) the key exchange is Diffie-Hellman. For RSA, the key exchange is key transport.

WTLS Handshake protocol: Session Resumption

If a given client and server have already performed a full handshake, subsequent handshakes may attempt to set up a secure connection by refering to the shared secrets previously agreed upon.

Such a handshake is called an “abbreviated handshake”, or “session resumption”. It requires less messages than the full handshake and does not require public key computations.

The PKI infrastructure (WPKI)

The use of certificates in the WTLS Handshake implies the existence of a Public Key Infrastructure. This is still an area of active development in the WAP forum, but version 1.0 of the WPKI specification is nearing completion.

Performance of WTLS

Performance analysis of the WTLS protocol must account for the different algorithms available. In the following we assume an anonymous Elliptic Curve Diffie-Hellman key exchange, the DES cipher and SHA-1-HMAC.

Bandwidth Analysis

The four messages exchanged between client and server in the full handshake have sizes

Description	Size in bytes
Client Hello	40
Server Hello, Server KE data	80
Client KE data, CCS, Finished	82
Server CCS, Finished	52

The size of each subsequent application-level datagram increases by no more than 33 bytes (due to the MAC tag, and block-cipher padding, mainly).

CPU usage

The following handshake times are for the Palm Pilot Palm IIIe plaform.

	Full Handshake	Session Resumption
CPU Time	2 seconds	<1 second
wireline modem	2.5 seconds	<1 second
CDPD wireless modem	3.5 seconds	<1 second

WTLS Performance notes

In client-authenticated RSA key agreement, the client must do an RSA signature, which is CPU-intensive.

In client-authenticated Diffie-Hellman, the client uses the public key in it's certificate for the key agreement; this computation is as fast as the anonymous case.

When either side is authenticated, and thus sends a certificate, the bandwidth goes up because the size of the certificate is always larger than the size of the raw public key. Although this area is still evolving, one expects server-side certificates to be on the order of 100 bytes. Client certificates will not be sent over the wire. They are stored in a database accessible by the server, and the client sends a reference to this certificate.

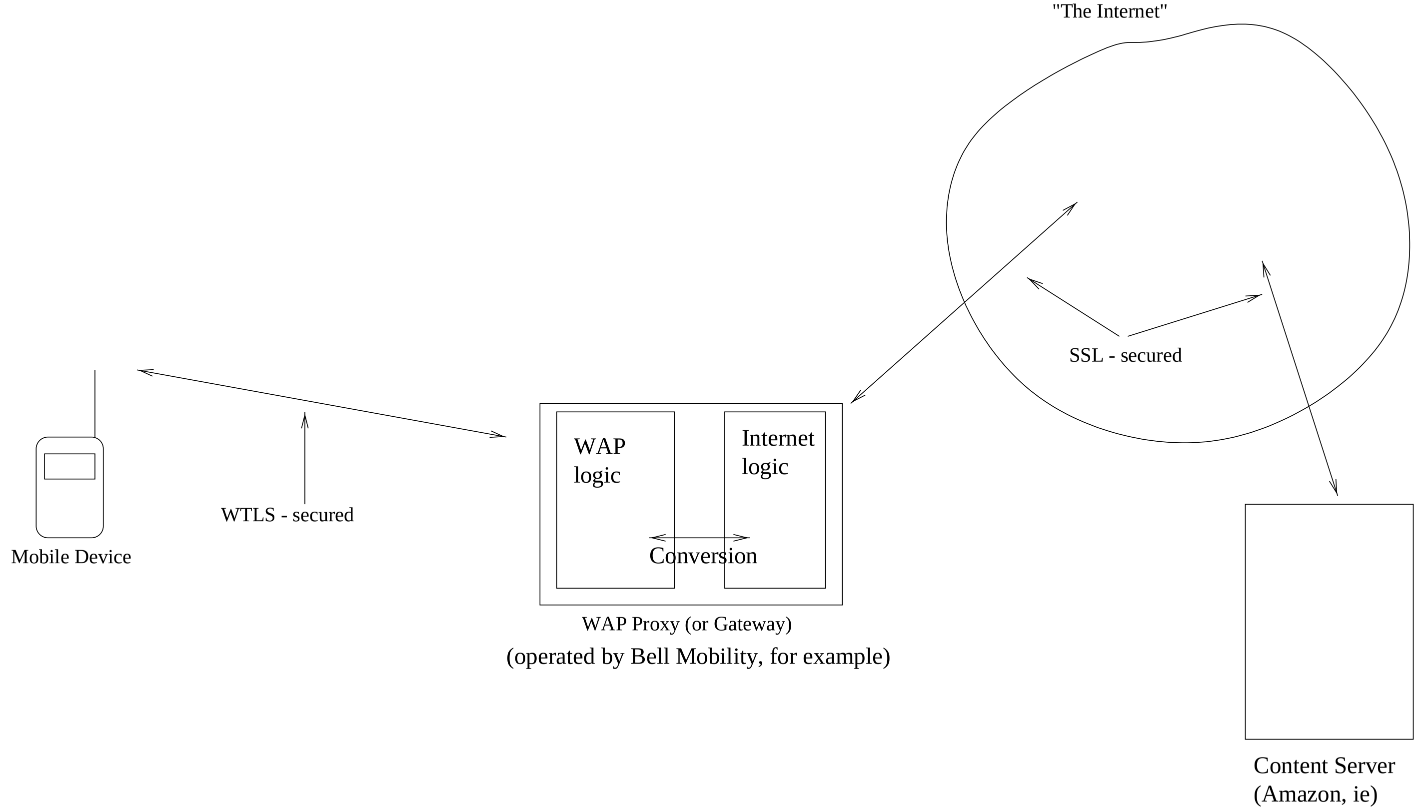

Figure 4: Security Architecture

Other Security mechanisms in WAP

There are some other security initiatives in WAP, such as

- SignText (WMLScript function for signing of text by the client device)
- WIM (Storage of your private cryptographic data on a token)
- E2E (An initiative to fix the hole between WTLS and SSL)

See http://www.wapforum.com for the relevant documentation.

Summary

The WTLS protocol is used in the transport layer security of the WAP protocol stack. It provides confidentiality and authentication services for datagrams.

The protocol is demonstratably efficient and is well suited to the characteristics of the wireless channel.