Comparison of WTLS and ITLS in Wireless end-to-end secure network (December 2002)

Younhee Kim, Chun-kit Wong, George Mason University

Abstract—End-to-end security is a major concern in current wireless network and business transactions. In this paper, we will study two end-to-end security supported protocols. An industry implemented security protocol, Wireless Transfer Layer Security (WTLS), and an academic proposed security protocol, Integrated Transfer Layer Security (ITLS) will be introduced. The current specification of WTLS does not provide total end-to-end security because WTLS-enabled gateway will leak plaintext during data transmission to the server. ITLS was created based on fixing WTLS security holes. A comparison of ITLS and WTLS demonstrates that ITLS provides stronger protection in gateway and offers a more secure channel than WTLS. Unlike in WTLS, where server trusts gateway, clients is the ITLS security partner of server in ITLS. All the encryption and decryption will be doubled on the client-side. Due to the limited resource on the client side (mobile devices), ITLS will perform slower than WTLS. We propose a modified ITLS that will increase ITLS performance in addition to providing the same security level as current ITLS. An implementation for WTLS is also introduced later in the paper.


1. INTRODUCTION

1.1 What is end – to – end security? Why do we need it?

End-to-end security is a concept that deals with securing data from an end-point to one end-point to another wherever it is created, stored, transmitted, accessed, or destroyed. End-to-end security plays a crucial role in the wireless network and business transactions. When a customer places an order with e-merchant, sensitive data is exchanged with the merchant, typically information including credit card number, delivery address, etc. A customer is less likely to engage in mobile commerce if there is a risk the privacy of the data will be violated anywhere in between the parties. Thus, Wireless Transfer Layer Security (WTLS) and Integrated Transfer Layer Security (ITLS) were created to address these security concerns.

In this paper, we will discuss in detail how WTLS works. In addition, ITLS will be introduced and compared with WTLS later on in the paper.

1.2 WTLS

To provide a secure wireless environment, Wireless Application Protocol (WAP) is a result of continuous work to define an industry-wide specification. Wireless Transfer Layer Security (WTLS) is part of security transfer protocol in WAP. WTLS provides authentication, privacy and data integrity between two applications communicating over a wireless network. WTLS is optimized for the relatively low-bandwidth and high-latency by incorporating features such as diagram support, streamlined protocol handshaking and dynamic key refreshing. Two protocols, handshake protocol and record protocol are explained in details in WTLS section of the paper.

1.3 ITLS

Integrated Transfer Layer Security (ITLS) is protocol proposed by Eun-Kyeong Kwon of Kaywon School of Art and Design, Yong-Gu Cho of Youngdong University, and Ki-Joon Chae of Ewha Womans University. The idea behind ITLS is to combine WTLS and TLS repetitive into one communication channel. The goal of ITLS is to eliminate what WTLS has encountered, particularly problems in WAP gateway. The security partner of a server is not a client via gateway but a client directly.

The rest of this paper is organized as follows. Section 2 describes the problem with WAP. Section 3 explains and compares two major protocols in end-to-end security: WTLS and ITLS. In section 4 we propose a modified version of ITLS. Section 5 talks about implementation of WTLS. Section 6 recapitulates the discussion of the paper.

2. THE PROBLEMS WITH WAP AND THE NEED OF END-TO-END SECURITY

In order to discuss how WTLS and ITLS works, WAP architectures and WAP data transmission will be introduced first.

2.1 WAP Architecture
In WAP, some security in communication between a WAP client and a content server is achieved if the HTTP communication between the WAP gateway and the content server is encrypted using TLS and if WTLS is used between the WAP client and the WAP gateway. However, during the data transmission, the data is decrypted into plaintext message in the WAP gateway and encrypted with another secret key to server. Illustrated in figure 2, WAP gateway has vulnerability where plaintext attack can be executed by intruders. Section 3.4.1 will describe in detail why WAP gateway presence may allows eavesdropping. Regardless of using WTLS and TLS/SSL, a malicious operator can eavesdrop and tamper with the data, which is a serious security. The other security threats include malicious code, lost or stolen devices.

3. COMPARISON OF THE TWO PROTOCOLS FOR END-TO-END SECURITY

In this section, we will thoroughly examine WTLS and ITLS as they enter their handshake protocol and record protocol.

3.1 Handshake protocol

Handshake protocol in WTLS is quite similar to TLS 1.0\(^*\). WTLS is used before any application starts the actual data transmission. Since certificates are an optional field in WTLS handshake protocol, this handshake protocol may allow the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in a WTLS record.

3.1.1 WTLS Handshake protocol

The following diagram shows the data flow inside WTLS handshake protocol.

![WTLS handshake protocol diagram](image)

To start the explanations of WTLS handshake protocol, we assumed that the gateway does not belong to the content server. Embedding a gateway in server is expensive, so the assumption fits the reality.

WTLS handshake protocol works very similar with Secure Socket Layer (SSL). In step 1 in figure 3, client first sends a ClientHello message to the gateway, this message includes session_ID, Random_Number, client_version, CipherSuite, Compression_method, sequence_num, key_refresh. CipherSuite gives a list of supported key exchange methods that the cryptographic keys for conventional encryption and MAC can use during the exchanged. WTLS handshake supports RSA, (fixed/ephemeral/anonymous) Diffie-Hellman and Fortezza key exchange methods. For more information regarding these key exchange methods, please refer to Appendix 1. CipherSuite parameters contain key exchange method and CipherSpec parameters that can be send to gateway. The following table will explain what elements CipherSuite has.

<table>
<thead>
<tr>
<th>CipherAlgorithm</th>
<th>MACAlgorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza MD5 or SHA-1</td>
<td></td>
</tr>
</tbody>
</table>
CipherType support: Block or Stream
HashSize: 0, 16 (for MD5), or 20 (for SHA-1) bytes
IV Size: Size of IV will be used for Cipher CBC mode

Table 1. WTLS cipherSuite parameter (key exchange method) from client

In step 2, Gateway gets Client_version, Random_Number, session_ID, CipherSuite and Compression_Methods. Then, gateway chooses key exchange method and other CipherSpec from the client’s list and sends server’s certificate to the client. Note that X.509 certificate and ServerKeyExchange are optional fields which depends on which key exchange method the gateway chooses. The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Hellman. A signature is created by taking hash of a message and encrypting it with gateway’s public key. The hash is defined as (1)

\[ \text{Hash}(\text{ClientHello.random} \ || \ \text{ServerHello.random} \ || \ \text{ServerParams}) \] (1)

Note that in (1), the Random_Number is generated by client containing 32-bit timestamp and 28 bytes generated by a secure random number generator. This Random_Number acts as nonce and will prevent replay attacks. A replay attack is one in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it. The ServerKeyExchange field is not required if Gateway used fixed Diffie-Hellman and RSA key exchange. When gateway chooses RSA key exchange, gateway will create a temporary RSA public/private key pair and use the ServerKeyExchange message to send the public key. Server may request for client’s certificate (CertificateRequest) if necessary. CertificateRequest contains certificate type (which indicates the public-key algorithm) and certificate authorities. At the end of step 2, gateway sends ServerHelloDone to client. The main goal of step 2 is to authenticate server and exchange gateway’s key.

In step 3, client generates premaster secret key/session key (ClientKeyExchange) to gateway. If server request client’s certificate in step 2, client will now send its Certificate to gateway. CertificateVerify message provides explicit verification of a client certificate. If gateway chooses certificate excluding fixed Diffie-Hellman parameters, CertificateVerify message will sign a hash code with either MD5 or SHA-1. At this point, client will also initiate ChangeCipherSpec message, which will include another new key algorithm CipherSpec to gateway. Client then sends Finished message with verified key exchange and authentication processes.

At step 4, gateway sends its own CertificateVerify and Finished message to client. In CertificateVerify message, gateway will transfer the pending to the current CipherSpec. Hence, WTLS handshake protocol is complete and both client and gateway can begin to transfer application data.

![Figure 3. WTLS handshake protocol in action](image-url)
3.1.2 ITLS Handshake protocol

Instead of communicating with two entities (gateway and client), ITLS will be handshaking with client, gateway and server. Since the security partner of a server is by client, several operations are added on the client-side. IntCertificate, Hash_Handshake and IntClientKeyExchange message. There will also be some modifications in gateway. The reason for adding these new operations will be explained in this section.

Looking at Figure 4, ITLS handshake protocol is somewhat similar to WTLS handshake protocol except ITLS deals with 3 parties and new operations are introduced during handshake communication. Note that some part of ITLS will not be explained explicitly as the information is identical to WTLS. It is advised to read section 3.3.1 first.

Table 2. New operations algorithm in ITLS handshake protocol

<table>
<thead>
<tr>
<th>Client</th>
<th>WAP G/ W</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientHello</td>
<td>ClientHello</td>
<td>ServerHello</td>
</tr>
<tr>
<td>Certificate*</td>
<td>IntCertificate*</td>
<td>ServerKeyExchange*</td>
</tr>
<tr>
<td>IntCertificate*</td>
<td>CertificateRequest*</td>
<td>Certificate</td>
</tr>
<tr>
<td>ServerHelloDone</td>
<td>ServerHelloDone</td>
<td>ServerHelloDone</td>
</tr>
<tr>
<td>Certificate*</td>
<td>Certificate*</td>
<td>Certificate*</td>
</tr>
<tr>
<td>ClientKeyExchange*</td>
<td>CertificateVerify*</td>
<td>CertificateVerify*</td>
</tr>
<tr>
<td>IntClientKeyExchange*</td>
<td>ChangeCipherSpec</td>
<td>ChangeCipherSpec</td>
</tr>
<tr>
<td>Finish</td>
<td>Finish</td>
<td>Finish</td>
</tr>
<tr>
<td>ChangeCipherSpec</td>
<td>ChangeCipherSpec</td>
<td>Finish</td>
</tr>
</tbody>
</table>

Note: * means optional field
Client will send a ClientHello message to gateway. This message includes session ID, random number, client version, CipherSuite, compression method, sequence num, key_refresh. The most important is to describe cipher suite. Similar to WTLS handshake protocol, client will give a list of key exchange methods CipherSuite and CipherSpec, including RSA, D/H, Fortezza, to gateway. All these information will be submitted to server. Client_key_ids, Trusted_key_ids, sequence_numbers and Key_refresh are kept inside gateway. For information of key exchange method, please refer to Appendix 1. Refer to table 2 for CipherSpec details.

Server gets Client_version and Random_Number and Session_ID and Cipher_suites and Compression_Methods from gateway. Server chooses key exchange method and other cipher spec from the list and send Server's certificate to the gateway. Optionally, server can request client's certificate (CertificateRequest). Note that if server chooses anonymous D/H, no Certificate will be sent by server side. If RSA is chosen as the key exchange method, server will generate temporary RSA key pair and sent temporary Server's public key (ServerKeyExchange) to the gateway. Server will send ServerHelloDone message at the end of this phase.

Gateway sends to the client all messages which it receives from server except IntCertificate. At this point, replay attack will be prevented by signature generation. The hash function of signature generation is defined as eq.1 earlier in WTLS.

\[
\text{ClientKeyExchange} = X = E_{\text{server's public key}}(SK_{cs})
\]

\[
\text{IntClientKeyExchange} = E_{\text{gateway's public key}}(\text{ClientKeyExchange} . SK_{cs})
\]

**Table 2. New operations algorithm in ITLS handshake protocol**

**Why ITLS needs IntCertificate?**

Referring to table 2, IntCertificate includes gateway’s public key. It is because client needs gateway’s public key in order to encrypt premaster secret key (ClientKeyExchange) between client and server. Note that this IntCertificate is computed from Certificate. ClientKeyExchange message is encrypted by server’s public key. Client generates another premaster key (IntClientKeyExchange) between client and gateway. IntClientKeyExchange message is encrypted by gateway’s public key. If before hand, the server requests certificate from Client (CertificateRequest), Client will send his own certificate to server which will be verified later on. At this point, client knows both gateway’s and server’s public key because client has Certificate and IntCertificate. Client will also initiate ChangeCipherSpec data, which will include another new algorithm and new CipherSpec to gateway.

Next, Hash_Handshake will be deliverered to client from gateway. This Hash_Handshake includes the hash of handshake_message between gateway & server. Hash_Handshake is defined as hash(handshake_message g,s)) The hash function and handshake_message have been declared and created by server earlier.

**Why ITLS needs hash_handshake?**

Hash Handshake provides a checksum mechanism between the client’s message to gateway and server’s message to gateway. This operation is helpful for ITLS Record protocol later on.

Client sends Finished message with verified hashed data which the gateway just sent. If gateway gets the finished message from client, gateway starts to send client certificate if it is requested and gateway decrypts IntClientKeyExchange and sends ClientKeyExchange to the server. Note that this ClientKeyExchange is computed from IntClientKeyExchange. Gateway also sends client's CertificateVerify and ChangeCipherSpec with the Finished message to the server. Server sends its own ChangeCipherSpec and transfers the pending data to the current ChangeCipherSpec. After server sends Finished message to gateway then to client, the ITLS handshake is complete.

In sum, the creation of optional IntCertificate, IntClientKeyExchange, and gateway’s Hash_Handshake protects the WAP gateway vulnerabilities in WTLS. These newly introduced operations eliminate the plaintext attack in WTLS. ITLS handshake protocol creates an environment for client to double encryption and decryption in ITLS Record protocol. This will be explained in detail in section 3.2.2.

### 3.2 Record protocol

Both WTLS and ITLS Record Protocol are layered protocols. The Record Protocol takes messages to be transmitted, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, and decompressed, then delivered to higher-level for processing. If a WTLS implementation receives a record type it does not understand, it ignores it.

#### 3.2.1 WTLS Record protocol

In figure 5, each entity has the ability to encrypt and decrypt the messages when messages reached to other entity. Note that only the communication between client and gateway is used for WTLS. The protocol used between gateway and server is TLS. The following table shows the elements contained in the messages.

**How does WTLS Record protocol work? And problems it created?**

<table>
<thead>
<tr>
<th>Type</th>
<th>Client → Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>SM_e=E(KE_c.gw,(SM_o+HMAC(MSc.gw,SM_o)))</td>
</tr>
<tr>
<td>Gateway</td>
<td>SM_o=D(KE_c.gw,SM_e)</td>
</tr>
</tbody>
</table>
As mentioned in section 2.3, WTLS cannot protect the decryption method applied. From server to client, the reverse direction of encryption and using gateway and server secret key. As for message sending (SMe') to server. Then server can decrypt message (SMe') between client and gateway. Gateway will carry on the similar message (SMo) with the hash function (HMAC) to original message (SMo) and MAC secret key (MSc, gw) of client and gateway. All these will create a new encrypted message called Sm. This message (Sm) will be decrypted in gateway using secret key between client and gateway. Gateway will carry on the similar encryption method as client and send the encrypted message (Sm') to server. Then server can decrypt message (Sm') using gateway and server secret key. For message sending from server to client, the reverse direction of encryption and decryption method is applied.

<table>
<thead>
<tr>
<th>Type:</th>
<th>Server → Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>RMo=D(EKc,gw,RMe)</td>
</tr>
<tr>
<td>Gateway</td>
<td>RMo=D(EKc,s,(RMo+HMAC(MSc,gw,RMo)))</td>
</tr>
<tr>
<td>Server</td>
<td>RMe=E(EKc,gw,(RMo+HMAC(MSgw,s,SMo)))</td>
</tr>
</tbody>
</table>

Table 3. WTLS Record Protocol data encryption and decryption

After WTLS handshake protocol, client and gateway successfully exchanges master key. Referring to table 3, from the message sent to the server, client will use the master key (EKc,gw) of client and gateway to encrypt application data (SMo) with the hash function (HMAC) to original message (SMo) and MAC secret key (MSc, gw) of client and gateway. All these will create a new encrypted message called Sm. This message (Sm) will be decrypted in gateway using secret key between client and gateway. Gateway will carry on the similar encryption method as client and send the encrypted message (Sm') to server. Then server can decrypt message (Sm') using gateway and server secret key. For message sending from server to client, the reverse direction of encryption and decryption method is applied.

<table>
<thead>
<tr>
<th>Client carries:</th>
<th>Gateway carries:</th>
<th>Server carries:</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKc,gw</td>
<td>EKc,gw</td>
<td>EKc,s</td>
</tr>
<tr>
<td>MS.c.gw</td>
<td>MSc,gw</td>
<td>MSc,s</td>
</tr>
<tr>
<td>EKc.s</td>
<td>EMS.gw</td>
<td>EMS,c.s</td>
</tr>
</tbody>
</table>

Table 4. Key distribution in WTLS Record Protocol

As mentioned in section 2.3, WTLS cannot protect the insecurity in WAP gateway. Table 4 shows the key each entity holds after WTLS handshake protocol. Since gateway has to use one key to decrypt the message (Sm) and another key to encrypt another message (Sm'), in between the encryption and decryption inside gateway, plaintext will be available for invader to attack. Table 4 shows a WAP gateway containing all the necessary key pairs to break all the messages in the session. To improve this insecurity, ITLS record protocol is introduced in next section.

3.2.2 ITLS Record protocol

As shown in figure 6, unlike WTLS Record protocol, ITLS creates double encryption and decryption on the client-side. Gateway can only decrypt messages from client and encrypt messages from server.

How does ITLS Record protocol work? And problems it created?

Table 5. ITLS Record Protocol data encryption and decryption

Table 5 shows the application data flow in ITLS record protocol. When ITLS handshake protocol completes, client, gateway and server successfully exchanges needed master keys. During the ITLS Record protocol, client does double encryptions. First, client will hash the message (SMo) using MAC secret key between client and server. This hashed value (SMo+HMAC(MSc,s,SMo)) is then encrypted with the original message (SMo) to form an encrypted message (Sm). Second, client will encrypt once more with encryption secret key(EKc,gw) between client and gateway. HMAC operation will be used before the second encryption. The second round encryption will create message (SMo2). Message (SMo2) will be transmitted to the gateway where the message (SMo2) will be decrypted by using encryption secret key (EKc,gw) between client and gateway. A one-encrypted-layer-left message (SMo) in gateway will transmit to server. Now server will decrypt (SMo) with encryption secret key (EKc,s) between client and server. As for message sent from server to client, the reverse direction of encryption and decryption method would be applied. Note that server will encrypt message (RMo) with HMAC function once using encryption secret key (EKc,s) between client and server. Gateway will encrypt message (RMo) with HMAC function on top of the previous server’s encryption using secret key (EKc,gw) between client and gateway. Client will receive double encrypted message (RMo2). Client will first decrypt message (RMo2) with (EKc,gw) and get message (RMo). Then, client decrypts (RMo) with (EKc,s) in order retrieve original message (RMo).

<table>
<thead>
<tr>
<th>Client carries:</th>
<th>Gateway carries:</th>
<th>Server carries:</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKc,gw</td>
<td>EKc,gw</td>
<td>EKc,s</td>
</tr>
<tr>
<td>EKc.s</td>
<td>MSc,gw</td>
<td>MSc,s</td>
</tr>
</tbody>
</table>

Table 6. Key distribution in ITLS Record Protocol

Comparing with Table 4 of Key distribution in WTLS Record Protocol, the above table 6 has fewer keys in gateway. In this section, based on the cryptographic keys client holds in ITLS record protocol, the concept of ITLS is to change the owner of the secret key, a client and server has each others secret key during one session. A client and gateway will share one secret key. Note that there are no changes to the server in...
ITLS record protocol. The main goal of ITLS is to eliminate the insecurity of WTLS-enabled gateway.

**Question raised in ITLS record protocol**

This section is created based on an interesting approach discovered during our team discussion analyzing the ITLS record protocol. Our team created another approach of ITLS record protocol as described below.

![Figure 7. Question on different approach to ITLS record protocol](image)

Why the current ITLS cannot operate double encryptions in server side when sending messages to client? It is not possible to encrypt twice in server side using the existing ITLS handshake protocol. The reason is that the server does not have the cryptographic secret key needed between server and gateway. Server only has one secret key (EKc,s) between client and server. In order to make this scenario (figure 7) work, our team has proposed a modified ITLS that will be explained in section 4.

### 3.3. Vulnerability (WTLS and ITLS)

The following chart provides the vulnerabilities that might happened in WTLS and ITLS, information based on Saarinen, Markku-Juhani. "Attacks against the WAP WTLS Protocol[7]."

<table>
<thead>
<tr>
<th>Case: Predictable IVs lead to chosen-plaintext attacks</th>
<th>WTLS</th>
<th>ITLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>The WTLS protocol's internal structure requires that packet information carry decipherable</td>
<td>ITLS will also suffer from the same case if the</td>
<td></td>
</tr>
</tbody>
</table>

For example, if the client and server agree to use DES for encryption, it takes 4.9 ms per blocks in WTLS while 9.8ms per block in ITLS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES encryption</td>
<td>4.9 ms/block</td>
</tr>
<tr>
<td>SHA-1</td>
<td>2.7 ms/block</td>
</tr>
<tr>
<td>512RSA key generation</td>
<td>3.4 minutes</td>
</tr>
<tr>
<td>512RSA sign generation</td>
<td>7028 ms</td>
</tr>
<tr>
<td>512RSA sign verification</td>
<td>438 ms (e=3)</td>
</tr>
<tr>
<td>163ECC-DSA key generation</td>
<td>597 ms</td>
</tr>
<tr>
<td>163ECC-DSA sign generation</td>
<td>776 ms</td>
</tr>
<tr>
<td>163ECC-DSA sign verification</td>
<td>2448 ms</td>
</tr>
</tbody>
</table>

*Table 8. Timing measurements for cryptographic algorithm on the PalmPilot*

### 3.4.2 Authentication

In ITLS, the client authenticates the server with server’s certificate and the gateway with gateway’s certificate during the handshake protocol. The server’s certificate is sent to the server chooses DES in CBC mode as the key exchange method to gateway and client.

<table>
<thead>
<tr>
<th>MTLS</th>
<th>ITLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTLS supports specific MACs (Media Access Controller) which do not ensure data integrity and is particularly weak when used in conjunction with stream ciphers.</td>
<td>ITLS will face the same problem as ITLS also support 40 bit XOR MAC.</td>
</tr>
<tr>
<td>WTLS will face the same problem as ITLS handshake protocol which contains exactly the same initial connection messages, ITLS may have similar problem in this case.</td>
<td></td>
</tr>
</tbody>
</table>

*Table 7. Vulnerability to WTLS and ITLS*

### 3.4.3 Confidentiality

In ITLS, the plaintext is never revealed in the gateway because client encrypts the message twice when it sends the message to the server. The gateway can have only the one-layered encrypted ciphertext and encrypt it again to send it to the server. When the server sends the message to the client, the server encrypts the message, sends to the gateway and the gateway re-encrypts over that cipher text. Therefore in ITLS the plaintext is never present in the gateway.

In WTLS, the gateway has the plain text because the gateway decrypts the cipher text first and then encrypt to the cipher between client and server. Therefore WTLS doesn’t provide the end-to-end confidentiality.
3.4.4. Integrity

In ITLS, the server sends the message to the gateway encrypted with the session key with client and server and the gateway encrypts it with the session key between the client and the gateway without decryption. This prevents the gateway from recognizing any errors or modifications the message was prone to while being transferred from the server. However, the last receiver, the client, can verify the authenticity of the original message from the server via the signature verification process if any errors occurred, the client can request the message to be re-send again or to stop the session.

In WTLS, the gateway can check for any modification in the received message because it can decrypt it and verify with a signature.

3.4.5 Summary of comparison

<table>
<thead>
<tr>
<th></th>
<th>WTLS</th>
<th>ITLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>T_in</td>
<td>T_in is about 2 * T_in</td>
</tr>
<tr>
<td>Authentication</td>
<td>Client authenticate the gateway. Gateway authenticate the server</td>
<td>Client authenticates the gateway and the server</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Secret key (client, gateway) Secret key (gateway, server) Plain text in the gateway</td>
<td>Secret key (client, server) Secret key (client, gateway) Still cipher text in the gateway</td>
</tr>
<tr>
<td>Integrity</td>
<td>Gateway can verify the modification in the middle. The end receivers can verify the modification</td>
<td>Gateway can’t verify the modification. The end receivers can verify the modification</td>
</tr>
</tbody>
</table>

Table 9. Summary of comparison

4. PROPOSAL OF THE MODIFIED ITLS

4.1 The reason ITLS needs to be modified

The ITLS client encrypts twice to send the message and decrypts twice to receive the message. This twice-processing causes more loads to the mobile limited resources and delay the transmission. We propose the modified ITLS to reduce the number of decryption from two to one.

4.2 The contents of the modification

We propose the establishment of the secret key between the gateway and the server. If the gateway and the server has the secure secret key, the server can encrypt twice the message to send the message and the gateway decrypt it in the middle and the end receiver, the client, can decrypt only once rather than twice.

Will this modified version be more vulnerable compared to the current ITLS and WTLS? It is more secure than WTLS. The reason is, even though the modified ITLS has the separate key between client, gateway and the secret key between gateway and server, the message is still encrypted with the secret key between the client and server.

Why is the performance of modified ITLS better than the current proposed ITLS? The modified ITLS puts a lower load in clients than the currently proposed ITLS. In addition, the modified ITLS provides the same secure environment as the currently proposed ITLS. The goal of the modified ITLS is to minimize the load on the mobile devices.

4.3 Modified Handshake Protocol

To have the secret key between the gateway and the server, the only modification of ITLS is that the message “GatewayKeyExchange” that gateway generates. The GatewayKeyExchange message encrypts the secret key between the gateway and the server with server’s public key. The gateway then sends it to the server with all other verified message.

GatewayKeyExchange = E(Pub_s, SK_g,s, others )

The gateway has the public key of server because the server sent it to the gateway with “Server Hello” in the early handshake. Figure 8 shows the modified handshake protocol of ITLS.

![Figure 8. Modified handshake protocol of ITLS](image-url)
4.4 Modified Record Protocol

Once the secret key is established between the gateway and the server, it is used for record protocol from the server to the client. Figure 10 shows the modified ITLS record protocol. When the server sends the message to the client through the gateway, it encrypts the message twice with the secret key between the client and server and between the gateway and the server. The gateway decrypts it and sends it to the client, client decrypts it only once and gets the original message the server has sent.

![Figure 9. Modified Record Protocol](image)

5. VENDER FOR WTLS IMPLEMENTATION: NOKIA®

NOKIA WAP phones does not support client authentication, however, NOKIA supports authentication of the WAP protocol gateway.

First, the terminal sends a message to the WAP protocol gateway indicating that security should be negotiated. The message contains a list of the algorithms that the terminal supports. The WAP protocol gateway chooses the algorithms in the secure connection, and may optionally return its certificate so that the terminal can check the identity of the WAP protocol gateway. The user will be notified that security has been accepted by the WAP protocol gateway.

If the terminal is unable to check the received gateway certificate, or if the certificate turns out to be invalid, information contained in the gateway certificate will be shown to the client. There are no NOKIA WAP phones with preinstalled CA certificates, so users must download CA certificates over the air. Terminal support certificates of the special WTLS type for which there has also been reserved a special MIME-type.

### 5.1 Example: Security features in NOKIA WAP Phone NOKIA 7110

<table>
<thead>
<tr>
<th>WTLS class support</th>
<th>WTLS Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported key exchanged algorithms</td>
<td>RSA no key size limit, 80bits MAC based on the SHA-1 algorithm, 160bits MAC based on the SHA-1 algorithm.</td>
</tr>
<tr>
<td>Supported bulk encryption</td>
<td>40bits MAC based on the SHA-1 algorithm, 80bits MAC based on the SHA-1 algorithm, 160bits MAC based on the SHA-1 algorithm.</td>
</tr>
</tbody>
</table>

| Certificate types | WTLS Root CA certificate. Must use ISO-Latin-1 character set |

#### Table 10. Security features in NOKIA WAP Phone

6. CONCLUSION

We described the importance of end-to-end security in current wireless network and explained two different protocols proposed by the industry and the academic field. WTLS is proposed by WAP and is implemented by industry while ITLS is proposed by the academic organization to eliminate the current WTLS protocol problem in which WAP gateway is opened to plaintext attack vulnerability.

We found ITLS is the protocol that avoids revealing the plaintext in the gateway by using two-layered encryption. However, encrypting twice causes more loads to the current limited resource mobile device and makes ITLS algorithm hard to be implemented.

We propose the modified ITLS protocol to reduce the number of decryption, which has the secret key between the gateway and the server. In this modified protocol, even though gateway decrypts the message with the secret key between the server and the gateway, the gateway still has the cipher text, which is different from WTLS, because the server encrypts the message twice.

ITLS is expected to be the good alternative protocol for WTLS if its modification goes in the direction of reducing times of client’s encryption and decryption.

7. APPENDIX

The following table describes the key exchange method used in WTLS/ITLS handshake protocol. The following information extracted from Stallings, William “Cryptography and Network Security”.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Security features in NOKIA WAP Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA:</td>
<td>The secret key is encrypted with the receiver’s RSA public key. A public-key certificate for the receiver’s key must be made available.</td>
</tr>
<tr>
<td>Fixed Diffie-Hellman:</td>
<td>This is a Diffie-Hellman key exchange in which the server’s certificate contains the Diffie-Hellman public parameters signed by the certificate authority (CA). That is, the public-key certificate contains the Diffie-Hellman public key parameters. The client provides its Diffie-Hellman public key parameters either in a certificate, if client authentication is required, or in a key exchange message.</td>
</tr>
<tr>
<td>Ephemeral Diffie-Hellman:</td>
<td>This technique is used to create ephemeral (temporary, one-time) secret keys. In this case, the Diffie-Hellman public keys are exchanged, signed using the sender’s private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys. This would appear to be the most secure of the three Diffie-Hellman options because it results in a temporary, authenticated key.</td>
</tr>
<tr>
<td>Anonymous:</td>
<td>The base Diffie-Hellman algorithm is used, with no</td>
</tr>
</tbody>
</table>
Diffie-Hellman: authentication. That is, each side sends its public Diffie-Hellman parameters to the other, with no authentication. This approach is vulnerable to man-in-the-middle attacks, in which the attack conducts anonymous Diffie-Hellman with both parties.

8. REFERENCE
