Performance Evaluation of Public-Key Cryptosystem Operations in WTLS Protocol

Albert Levi
Sabanci University
Istanbul, Turkey
levi@sabanciuniv.edu

Erkay Savas
Sabanci University
Istanbul, Turkey
erkays@sabanciuniv.edu

Abstract

WTLS (Wireless Transport Layer Security) is an important standard protocol for secure wireless access to Internet services. WTLS employs public-key cryptosystems during the handshake between mobile client and WAP gateway (server). Several cryptosystems at different key strengths can be used in WTLS. The trade-off is security versus processing and transmission time. In this paper, an analytical performance model for public-key cryptosystem operations in WTLS protocol is developed. Different handshake protocols, different cryptosystems and key sizes are considered. Public-key cryptosystems are implemented using state-of-the–art performance improvement techniques, yielding actual performance figures for individual cryptosystems. These figures and the analytical model are used to calculate the cost of using public-key cryptosystems in WTLS. Results for different cryptosystems and handshake protocols are comparatively depicted and interpreted. It has been observed that ECC (Elliptic Curve Cryptography) performs better than its rival RSA cryptosystem in WTLS. Performance of some stronger ECC curves, which are not considered in WTLS standard, is also analyzed. Results showed that some of those curves could be used in WTLS for high security applications with an acceptable degradation in performance.

1. Introduction

The extensive use of mobile communication has created an important demand for value-added services. WAP (Wireless Application Protocol) [1] is a framework for developing applications to run over wireless networks. WAP is developed by an international industry-wide organization called the WAP Forum.

WTLS (Wireless Transport Layer Security) [2] is the security protocol of the WAP protocol suite. WTLS operates over the transport layer and provides end-to-end security, where one end is the mobile client, and the other end is the WAP gateway. WAP gateway acts as a proxy of the mobile client to access an application server hosted somewhere on the Internet. The communication beyond the WAP gateway is conducted using the regular Internet (TCP/IP) protocol suite.

A set of handshake messages is exchanged in order to set up a secure environment between the mobile client and the server (WAP gateway). Cryptographic algorithms, keys and related parameters are negotiated during the handshake. Once the handshake messages are exchanged and session key is generated, all WTLS and upper layer protocol messages can be exchanged in encrypted form. In this way, confidentiality and integrity are provided.

Authentication is an optional service in WTLS. Authentication is provided if the parties provide *digital certificates* during the handshake. Certificates are digital identities that contain public-keys to be used during the key exchange. Certificates are issued by trusted Certification Authorities (CA) with a digital signature on the certificate content. Validation of a certificate means the legitimacy of the enclosed public-key. A party, who does not have a certificate, should use an unapproved public-key. Therefore, that party cannot be authenticated.

Certificate validation, authentication and session key exchange use asymmetric public-key cryptosystems that require computation-intensive processes, and are therefore slow. Speed is inversely proportional to the key size used in public-key cryptosystems. Since the processing power of mobile clients is limited, relatively smaller keys are selected for WTLS. Moreover, data transfer rate is also limited in mobile communication environment and using smaller keys would help to save bandwidth.

1.1. WTLS performance evaluation in the literature

Apostolopoulos, et al. have given performance evaluation studies for the TLS protocol, which is considered as the ancestor of WTLS, in [8] and [9]. The primary aim of those studies is to measure server-side throughput and latency versus the number of secure HTTP requests. The measurements are performed in a testbed environment.

The important difference between TLS and WTLS performance evaluations is that the client-side performance criteria are more important in WTLS than in

TLS. Although the protocols are similar, WTLS exercises some precautions because of low processing power of the client-side wireless equipments and low data transfer rates of the wireless medium. Examples to those precautions are use of short key sizes for the cryptographic primitives and removing optional fields in the public key certificates.

Herwono and Liebhardt have given two simulation-based WTLS performance evaluation studies in [10, 11]. Both server and client routines are simulated in a single computer. Those studies give encryption/decryption and handshake timings, but only for standard cryptosystems and key-exchange suites specified in WTLS specification. No alternative cryptosystems are analyzed. Moreover, server-side queuing delays are not considered.

1.2. Contribution of the paper

The desire to use public-key cryptosystems with smaller key sizes for performance reasons is understandable for WTLS. However, the effect of public-key cryptosystem selection in the performance should be analyzed in order to see the trade-off. We believe that such an analysis can be done analytically since WTLS has a smooth protocol run. Our contribution in this paper is an analytical performance evaluation for public-key cryptosystem operations in WTLS. Two practical and secure handshake protocols of WTLS are analyzed. The performance figures are latency, processing time and amount of data transmitted due to public-key cryptosystem operations. We also implement important public-key cryptosystems. Some of those cryptosystems are proposed in WTLS standard, some are our own candidates with larger key sizes. We used timings obtained from those implementations in our analyses. Results are graphically shown and interpreted.

The rest of the paper is organized as follows. Section 2 briefly discusses the use public-key cryptosystems in WTLS standard. The handshake protocols are detailed in Section 3. Performance analyses are given in 4. Section 5 summarizes conclusions reached by this study.

2. Public-key cryptosystems in WTLS

Public-key cryptosystem operations use two different, but related keys: public-key and private-key. Public-key operations are for encryption and signature verification. Private-key operations are for decryption and signature issuance. Key exchange operations are also public-key cryptosystem operations, but their nature depends on the cryptosystem used.

Public-key cryptosystems are used in the WTLS handshake for key exchange and certificate verification purposes. Authentication is automatically provided when key exchange is performed using certified keys. WTLS supports two public-key cryptosystems: RSA (Rivest-Shamir-Adleman) [3] and ECC (Elliptic Curve Cryptography) [4].

2.1. Public-key cryptosystems for key exchange and certificate verification

If RSA is to be used for key exchange, the encryption/decryption feature of RSA is employed. If ECC is to be used, ECDH (Elliptic Curve Diffie-Hellman) [5] key exchange method is employed.

Regular DH (Diffie-Hellman) [6] method is proposed as another key exchange mechanism in WTLS standard. However, the standard proposes DH method only for completely anonymous handshakes, in which neither client nor server use certificates to authenticate themselves. Anonymous handshakes are vulnerable to man-in-the-middle-attacks, where an adversary impersonates both parties. Therefore, we do not consider anonymous handshakes as secure methods and do not include them in our performance evaluation. Besides DH, WTLS also proposes anonymous versions of RSA and ECDH methods that we disregard as well.

Certificate verification is a public-key operation. Both RSA and ECC can be used. If ECC is to be used, ECDSA (Elliptic Curve Digital Signature Algorithm) [7] is employed. If RSA is to be used, its verification feature is employed. Certificate generation, which is signature issuance, is not a part of WTLS protocol.

2.2. Key exchange suites of WTLS

WTLS uses the term *key exchange suite* to specify the public-key cryptosystem pair to be used for certificate validation and key exchange. WTLS supports several alternative key exchange suites. However, only two of them offer an acceptable level of security: ECDH_ECDSA and RSA key exchange suites.

1. **ECDH_ECDSA:** ECDSA is used for certificate verification. Certificates include ECDH parameters to be used for key exchange.
2. **RSA:** RSA cryptosystem is used for both certificate verification and key exchange.

3. WTLS handshake protocols

WTLS standard [2] proposes different handshake alternatives. A relatively faster handshake protocol, known as “abbreviated handshake”, is possible when client and server are willing to resume a previous secure connection without any public-key cryptosystem operation. This is a performance improvement aspect of WTLS. However, this approach does not work when the client and server meet for the first time or the when their old session expires. Eventually, client and server should run a *full handshake*. We deal with two practical full

handshake protocols of WTLS in this paper: (1) with server-only authentication, (2) with mutual authentication.

3.1. Protocol 1: Full handshake with server-only authentication

The client is not authenticated in this protocol. Only the server is authenticated using certificate. The protocol is depicted in Figure 1.

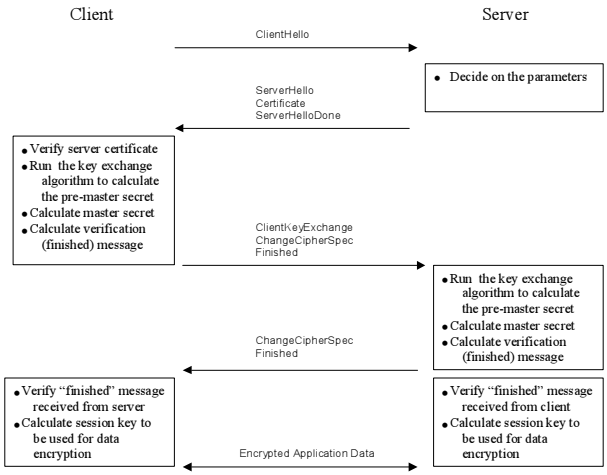


Figure 1. Full Handshake with server-only authentication

Server's certificate includes either server's RSA public-key (if RSA is used for key exchange) or ECDH parameters (if ECDH is used for key exchange). The certificate is digitally signed by a trusted CA. Upon receipt of the server's certificate, the client verifies the signature over the certificate to learn the server's ECDH parameters or RSA public-key to be used in key exchange. "ClientKeyExchange" message is used for the client to send its key exchange data to server. This data is either the client's ECDH parameters or an RSA encrypted message, depending on the key exchange mechanisms agreed by the hello messages. If RSA encrypted message is sent, the server should decrypt it. Rest of the protocol does not use any public key cryptosystem operations.

In this protocol, the client does not send its public key exchange data in a certificate; therefore the client cannot be authenticated by the server.

3.2. Protocol 2: Full handshake with mutual authentication

This protocol incorporates client certificates as well. In this way, mutual authentication is established between client and server. The protocol is depicted in Figure 2.

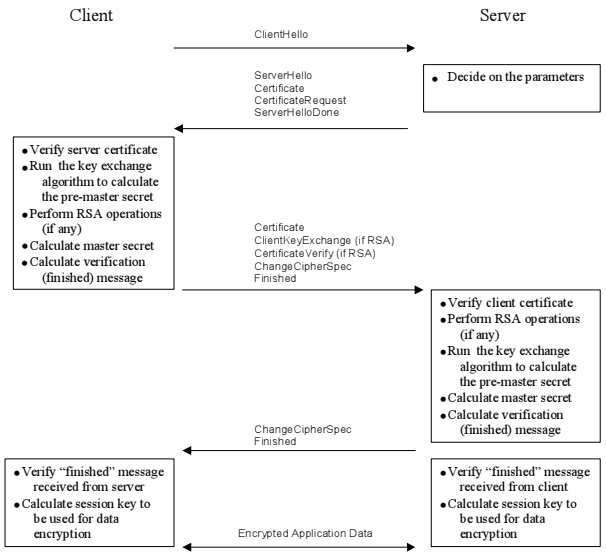


Figure 2. Full Handshake with mutual authentication

This protocol is similar to the previous one except for the extra processing for the client's certificate. The server requests the client's certificate after sending its certificate. In response, the client sends its certificate. The server validates this certificate. If they are using RSA cryptosystem, "ClientKeyExchange" and "CertificateVerify" messages should follow the client certificate. The ClientKeyExchange message costs an extra RSA encryption to the client, and an RSA decryption to the server. The CertificateVerify message costs an RSA signature to the client, and an RSA signature verification to the server. ClientKeyExchange message contains the pre-master secret. CertificateVerify message is for authentication purposes.

If ECDH is being used, "ClientKeyExchange" and "CertificateVerify" messages are not sent. The client's public ECDH parameters are sent in the client's certificate. Each party runs the ECDH algorithm using its own private-key and other's public ECDH parameter to calculate the pre-master secret.

4. Performance evaluation

In this section, performance evaluation of the public-key cryptosystem operations in WTLS handshake protocols is given. We have considered the *additional cost* associated with public-key cryptography in WTLS. Three cost factors are examined: (i) the processing time, (ii) the amount of data produced and transmitted, (iii) response time (latency) as seen by the client.

Both handshake protocols described in Sections 3.1 and 3.2 are formulated separately. Formulations vary depending on the key exchange suit. ECDH_ECDSA and RSA key exchange suites, described in Section 2, are

considered for each protocol. Formulations could not be given in this paper due to space limitations.

Moreover, we have implemented ECC and RSA cryptosystem operations in software using state-of-the-art computational techniques. Client implementations are performed on ARM7TDMI 25MHz microprocessor, which is a popular one for mobile phones and PDAs. Server platform is a Pentium II 450 MHz. Table 1 demonstrates approximate comparable key sizes for ECC and RSA cryptosystems. We implemented ECDSA [7] and ECDH [5] algorithms for six different predefined elliptic curves shown in Table 1, as levels 1 and 2. Those curves are recommended in the WTLS standard [2]. We also included implementations for three additional curves (namely 256P, 283K and 283R in Table 1) recommended in [7], which provide much higher security to give an idea about the cost of using more secure curves than those recommended in WTLS standard.

Table 1. Cryptographic strength of RSA and ECC

Strength Level	ECC	RSA
1	160P, 163K, 163R	1024
2	224P, 233K, 233R	2048
3	256P, 283K, 283R	3072

Finally, we have taken timings of the cryptosystems of Table 1 and used them to sketch the performance of public-key cryptography in WTLS handshake protocols. These analyses are given in Sections 4.1 and 4.2.

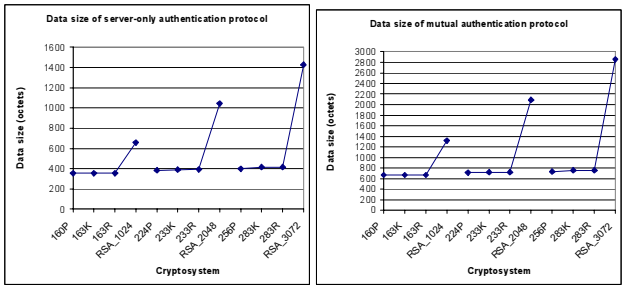
Assuming that a typical server handles requests from many clients in a given unit time, we run the cryptographic algorithms repeatedly many times on server machine and take the arithmetic mean of the measured times. On the other hand, since a client device run cryptographic algorithms occasionally, the measurement is performed on a cold cache (i.e. the code is not in the cache) for client device. Every measurement is done several times separately and the final timing is obtained using the arithmetic mean. While the variations on the measurement on client device are small as we anticipated, the variations for the first several runs are considerably high on the server machine because of the cache effect.

4.1. Client and server performance analyses

Public-key cryptosystem related data transfer amount and processing time are analyzed for two WTLS handshake protocols given in Section 3.

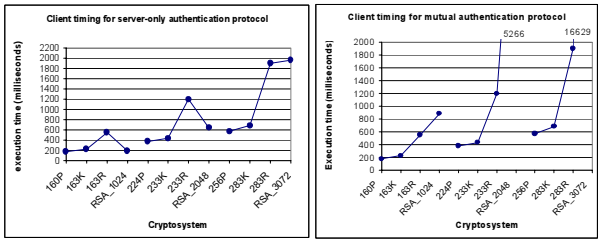
WTLS standard [2] proposes WTLS certificates, which is optimized in size. Although some alternatives exist, WTLS certificates are the most suitable ones because of their size advantage. Therefore, WTLS certificates are assumed in our calculations. Although the WTLS standard allows using certificate chains with several certificates on it, we assumed single certificate per certificate chain in our calculations. This is a practical necessity, especially for the server certificate chain, to reduce the overhead on mobile client.

Performance figures versus cryptosystems are depicted in Figure 3, Figure 4 and Figure 5. Client and server timings are shown in different figures for each protocol. In each figure, cryptosystems are grouped according to their security levels given in Table 1.



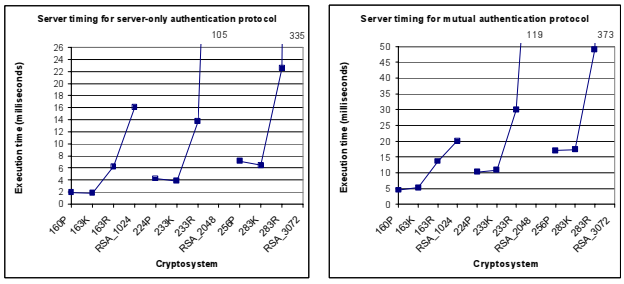
(a) (b)

Figure 3. Amount of public-key cryptosystem related data transmitted (a) in the handshake protocol that supports server authentication only (protocol 1), (b) in the handshake protocol that supports mutual authentication (protocol 2)



(a) (b)

Figure 4. Client's processing time of public-key cryptosystem operations (a) in the handshake protocol that supports server authentication only (protocol 1), (b) in the handshake protocol that supports mutual authentication (protocol 2)



(a) (b)

Figure 5. Server's processing time of public-key cryptosystem operations (a) in the handshake protocol with server-only authentication (protocol 1), (b) in the handshake protocol with mutual authentication (protocol 2)

We analyzed those figures to evaluate the effect of cryptosystem choice on the performance of WTLS, and obtained the following results.

1. ECC has a best curve option for each security level and handshake protocol. In other words, ECC is better than RSA in WTLS. The "R" family of curves (163R, 233R and 283R) performs worse than other curves. Thus, the best curve option is either "P" (160P, 224P, 256P) curves or "K" (163K, 233K, 283K) curves.
2. The performance of RSA is generally poor. For some cases, it is unacceptably slow (e.g. RSA_2048 and RSA_3072 in Figure 4b, Figure 5a and b). RSA_1024 and 2048 have reasonable client timing figures for server-only authentication protocol (Figure 4a), but the peer timing figures for the server (Figure 5a) are slow. Consistency between the client and the server timing figures for the same protocol and cryptosystem is very important, because the client and the server should use the same protocol and the same cryptosystem. If one cryptosystem works well for the client, but bad for the server, then a performance conflict occurs. RSA_1024 and RSA_2048 cause such a conflict for the handshake protocol with server-only authentication.
3. A similar performance conflict between the client and server is valid for ECC too. For example, the best level 2 curve for client in server-only authentication protocol is 224P (Figure 4a), but it is 233K for the server (Figure 5a). However, the level of conflict in ECC is not so severe since the performances of the conflicting curves are close to each other.
4. RSA produces more data to be exchanged than ECC as can be seen in Figure 3a and b. In other words, a WTLS handshake using RSA requires more transmission time than using ECC. This is another advantage of ECC over RSA. This fact is valid for all security levels and protocols.
5. Mutual authentication has an extra processing burden on top of server-only authentication. Fortunately, this is not an additional overhead for client as long as ECC curves are selected. This fact can be visualized by comparing Figure 4a and Figure 4b. This is good news since mobile clients have limited processing power. The burden of mutual authentication is on the server (compare Figure 5a and Figure 5b). Moreover, mutual authentication almost doubles the data exchanged (compare Figure 3a and Figure 3b).
6. We analyzed the performance of level 3 curves (256P, 283K and 283R) that are not proposed in the WTLS standard. Figure 4 and Figure 5 show that 283R curve is not so useful since it is very slow. The same figures also show that 256P and 283K curves perform worse than smaller curves, but their timings are in within acceptable limits. For example, the client's processing time for 256P curve is 570 milliseconds; this value is 376 milliseconds for 224P curve. Such overhead may be preferred by security and privacy sensitive people and applications.

4.2. Response time analysis

Above discussions and Figure 3, Figure 4 and Figure 5 provide necessary input to calculate the cryptographic burden on the additional WTLS response time due to public-key cryptosystem operations. This response time is the sum of client and server processing times, data transmission time and server-side waiting time. Waiting time is important for only the server-side since the same server is expected to provide service to several WTLS connections in parallel, but this is not the case for the client. We model the server as an M/M/1 queue, so the average waiting time, T_w, is given as $T_w = T_s.\rho / (1-\rho)$, where ρ represents the utilization of the server and T_s represents the server-side processing time per request.

The data transmission time, T_{data}, is simply computed as $T_{data} = 8L / R$, where R is the channel transmission rate in bps and L is the data length in octets. WTLS response time due to public-key cryptosystem operations, T_{resp}, is calculated as $T_{resp} = T_c + T_s + T_w + T_{data}$, where T_c represents the client side processing time.

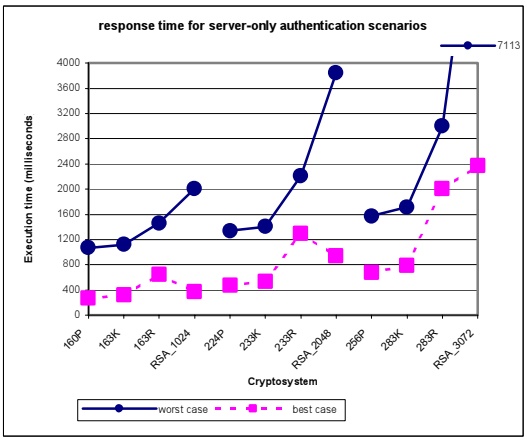
Response time values for different cryptosystems are shown in Figure 6. Both server-only and mutual authentication protocols are considered. For each protocol, worst and best server utilization and communication speed scenarios are analyzed. Worst case corresponds long server waiting time (i.e. high server utilization) and low communication rate. In our analysis, ρ is taken as 0.8 and R is taken as 6 Kbps for the worst case. For the best case, we assumed negligible server waiting time and 56 Kbps wireless communication rate.

As can be see from Figure 6a, RSA suffers from large variance in server-only authentication protocol. That means network and server conditions may significantly alter RSA's performance. Variation in response time is not a problem for RSA in mutual authentication protocol (Figure 6b), but RSA is the worst option in all three security levels.

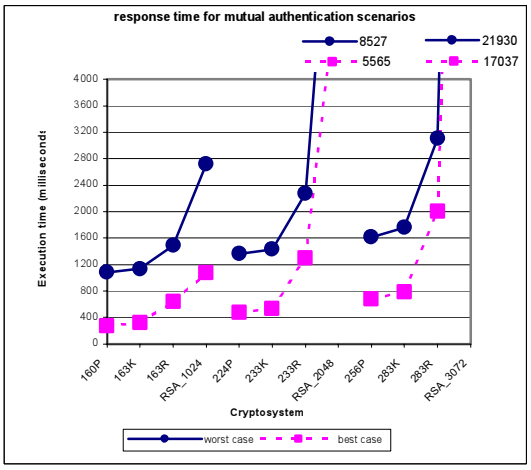
5. Conclusions

WTLS (Wireless Transport Layer Security) [2] is a security protocol between a server and a mobile client with limited processing power. Different public key cryptosystems with different key sizes have different performance characteristics in WTLS. In this paper, we employed analytical techniques to analyze the performance of public-key cryptosystem operations in WTLS. The performance figures are latency (response time), processing time and the amount of data transmitted between the client and the server. Two WTLS handshake

protocols are considered: server-only authentication and mutual authentication.



(a)



(b)

Figure 6. Worst and best response times due public-key cryptosystem operations (a) in the handshake protocol with server-only authentication (protocol 1), (b) in the handshake protocol with mutual authentication (protocol 2)

Different cryptosystems are employed. These are RSA with 1024, 2048 and 3072 bit key (modulus) sizes, and several ECC curves. Some of those curves are standard WTLS curves; some are our candidate curves that are cryptographically stronger. We implemented these cryptosystems using state-of-the-art performance improvement techniques. Timings obtained from our implementations are used in the analytical model to get the actual WTLS performance figures for public-key cryptosystem operations. Those figures are graphically analyzed. Results showed that, comparing processing time and the amount of data exchanged between the client and the server, ECC curves perform better than the RSA cryptosystems in WTLS. Especially, 2048 and 3072 bit RSA should not be used in WTLS for performance reasons. 1024 bit RSA has a decent performance at client, but not at server as compared to the performance of ECC curves. Moreover, RSA experiences an important variation in response time for server-only authentication protocol: for high-speed connections and negligible server-side waiting time, RSA performs relatively well, but under low speed connections and high server utilization, RSA response time is high.

We also analyzed the performance of stronger ECC curves that are not proposed in the WTLS standard. Two of those curves, namely 256P and 283K curves, performed within acceptable limits and can be used for high-security applications in WTLS.

6. References

1. WAP Forum, Wireless Application Protocol Architecture Specification, WAP-210-WAPArch-200100712-a, 12-July-2001 version, latest version is available at http://www.wapforum.com.
2. WAP Forum, Wireless Transport Layer Security Specification, WAP-261-WTLS-20010406-a, 06-April-2001 version, latest version is available at www.wapforum.com.
3. R. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978.
4. N. Koblitz, Elliptic *curve cryptosystems*, Mathematics of Computation, 48(177):203 -209, January 1987.
5. IEEE Standard specifications for public-key cryptography, IEEE Std. 1363-2000, 30 January 2000.
6. W. Diffie, and M. E. Hellman, *New directions in cryptography*. IEEE Transactions on Information Theory, 22:644-654, November 1976.
7. National Institute for Standards and Technology, Digital Signature Standard (DSS), FIPS PUB 186-2, January 2000.
8. G. Apostolopoulos, V. Peris, and D. Saha, "Transport Layer Security: How much does it really cost?," Proc. IEEE Infocom, March 1999.
9. G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, Securing Electronic Commerce: Reducing the SSL Overhead, IEEE Network, (14)4: 8-16, July/Aug 2000.
10. I. Herwono, and I. Liebhardt, "Performance Evaluation of the WAP Security Protocols," Proceedings of the 10th Aachen Symposium on Signal Theory, Sept. 2001, pp. 95 –100, Aachen, Germany. Available from http://www.comnets.rwth-aachen.de
11. I. Herwono, and I. Liebhardt, "Performance of WTLS and Its Impact on an M-Commerce Transaction," ICICS 2001 - Proceedings of the Third International Conference on Information and Communications Security, Nov. 2001, pp. 167 – 171, Xi'an, China. Available from http://www.comnets.rwth-aachen.de