Cryptographic Execution Time for WTLS Handshakes on Palm OS Devices

Neil Daswani
Stanford University
daswani@cs.stanford.edu

Abstract

This paper analyzes the cryptographic operation time that is required to execute secure transactions on wireless PDAs with WAP browsers. We evaluate the time required to execute the necessary cryptographic operations to set up a WTLS connection on a Palm OS device with both ECC-based public key cryptography as well as with RSA-based public key cryptography. We find that the execution times for server-authenticated 1024-bit RSA handshakes can be up to twice as fast as for server-authenticated 163-bit ECC-based handshakes, but that the execution time for mutually-authenticated (client and server authenticated) handshakes is at least eight times faster using ECC-based handshakes.

1. Introduction

WAP browsers for wireless handhelds such as Palm, Windows PocketPC, Psion, and RIM devices are gaining popularity, and may prove to become an important part of how users access wireless services. For example, AU Systems [3] has ported its WAP browser to the Palm platform. EZOS [4] has developed EzWAP, a WAP browser that supports the Windows CE platform. Psion [5] has licensed WAP browser technology from Purple Software and Dynamical Systems Research, and is currently beta testing the product. In addition, Neomar [6] has built a WAP browser for the RIM Interactive Pager platform. In the near future, WAP browsers may become a standard "in-the-box" application on wireless PDAs.

As of the writing of this paper, none of these WAP browsers support secure connections or transactions with the gateways that they communicate with, although some companies have announced that such support is forthcoming. Neomar, for example, has licensed WTLS Plus from Certicom [7], and will be incorporating WTLS support in upcoming versions of its WAP browser on the Palm and RIM platforms. If mobile commerce is to take-off, WAP browsers will be required to make secure connections to the WAP gateways with which they communicate. Creating a secure connection will incur some computational overhead, and will likely increase the time required to set up a connection with a gateway.

It is important to understand exactly how much extra time setting up a secure connection will require. If too much time is required to set up a secure connection, this may affect the usability of the wireless PDA for secure transactions, and users may decide that executing secure transactions on wireless PDAs is simply too inconvenient. At the same time, if the appropriate level of security is not provided, corporations may not be willing to take the risk to allow electronic commerce transactions from mobile devices.

This paper analyzes the cryptographic operation time that is required to execute secure transactions on wireless PDAs with WAP browsers. We evaluate the time required to execute

the necessary cryptographic operations to set up a WTLS connection on a Palm OS device with both ECC-based public key cryptography as well as with RSA-based public key cryptography. We find that the execution times for server-authenticated 1024-bit RSA handshakes can be up to twice as fast as for server-authenticated 163-bit ECC-based handshakes, but that the execution time for mutually-authenticated (client and server authenticated) handshakes is at least eight times faster using ECC-based handshakes.

Section 2 of this paper reviews WTLS and the cryptographic computations that must be computed by the client (in our case, a wireless PDA). Section 3 describes the results of experiments benchmarking various cryptographic operations on a Palm VII device, which will serve as a prototypical wireless PDA for our study. Using the benchmarking results obtained in Section 3, Section 4 estimates the time required to execute secure handshakes in both server-authenticated and mutual-authenticated handshakes. Section 5 comments on our results, and Section 6 concludes the paper.

2. A Review of WTLS

WTLS is the Wireless Transport Layer Security protocol designed to support the security requirements of authentication, privacy, and integrity in the Wireless Application Protocol (WAP) [1] defined by the WAP Forum.

In the following, we review the steps required in a full WTLS handshake, and we will identify the necessary cryptographic operations required on the client. Although this section is written to be self-contained, the reader is referred to [2] for an overview of WTLS.

In a full WTLS handshake, two round-trips are required before the client and server can start exchanging encrypted application data. In the first round-trip, client and server hello messages are exchanged. These messages are used to negotiate protocol versions, key exchange suites, cipher suites, and a number of other parameters. If the client is to authenticate the server, the server is expected to send the client its public-key certificate in the second half of the first round trip. When the client receives the server's certificate, it validates the certificate by verifying the CA's signature on the certificate.

The exact messages that are exchanged next depend upon 1) whether only the server or both parties are to be authenticated, and 2) whether RSA or ECC is to be used for key agreement and authentication. Let us consider the server-authenticated only case first. The reader may refer to Figure 1 for an illustration of the messages that are exchanged between the client and server in the server-authenticated only case.

Server-Authenticated Only

In the server-authenticated only case, the client sends a `ClientKeyExchange` message to the server after receiving the server's certificate.

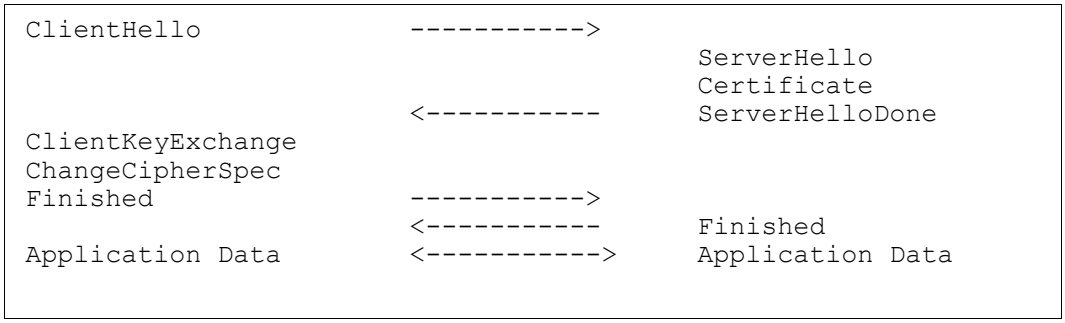

Figure 1: Server-Authenticated Only WTLS Messages

If RSA is to be used for key agreement, the client generates a random value to be used as the pre-master secret, and encrypts the pre-master secret with the server's public key. The encrypted pre-master secret is sent to the server in the `ClientKeyExchange` message. The server decrypts the pre-master with its private key, and the pre-master secret is now known to both parties.

On the other hand, if ECC-DH is to be used for key agreement, the client generates an EC Diffie-Hellman public key, and sends it to the server in the `ClientKeyExchange` message. To derive the pre-master secret, the client multiplies the server's public key with its the Diffie-Hellman private key. (The server derives the pre-master secret by multiplying the EC Diffie-Hellman pubic key with its private key.)

Therefore, in the server-authenticated only case, the cryptographic operations required on the client are: 1) verification of the server's certificate, and 2) session key establishment. In the case that RSA is used for key agreement, an encryption with the server's public key is required. In the case that ECC-DH is used for key agreement, the client needs to generate an ECC-DH key pair and the appropriate multiplications for key agreement need to take place.

Mutual Authentication

The messages exchanged in the mutual-authentication case are shown in Figure 2. After the `ServerHelloDone` message is received, the client sends its certificate to the server.

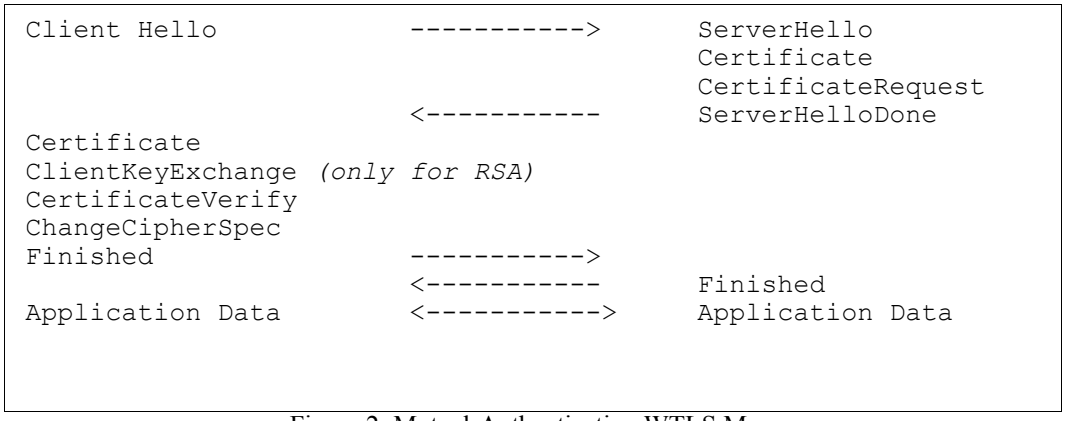

Figure 2: Mutual-Authentication WTLS Messages

If RSA is to be used for key agreement, the client transmits a `ClientKeyExchange` message to the server just as in the server-authentication only case, but then sends a `CertificateVerify` message to the server containing an RSA signature on all handshake messages starting from the `ClientHello` message up to (but not including) the `CertificateVerify` message.

If ECC-DH is to be used for key agreement in the mutual-authentication case, a `ClientKeyExchange` message does not need to be sent to the server since we assume the client's certificate (that was already sent to the server) contains the appropriate ECC-DH public key. However, an explicit ECC-DSA signature is required to authenticate the client, and this signature is sent in a `CertificateVerify` message.

The cryptographic operations required on the client in the mutual-authentication case are the same as those in the server-authenticated only case with the addition that the appropriate RSA or ECC-DSA signature needs to take place to create the `CertificateVerify` message.

After the appropriate `Certificate` and `ClientKeyExchange` messages are sent by the client as needed, the pre-master secret is set, the master secret is computed, the client transmits the `ChangeCipherSpec` message, both parties transmit `Finished` messages, and the client and server can then start exchanging application data encrypted with the master secret. The computation of the master secret from the pre-master secret only requires a number of hashes whose computation time is relatively insignificant compared to the other cryptographic operations required of the client, and hence will not be explicitly included in the timing estimates described in this paper.

3. Benchmarking Experiments

Figure 3 shows execution times for various ECC and RSA cryptographic primitives on various Palm OS platforms.

	New Palm VII (Dragonball-EZ, 20MHz, PalmOS v.3.2.5) (ms)	Palm V (Dragonball-EZ, 16.6MHz, PalmOS v.3.3) (ms)	Old Palm VII (Dragonball, 16.6MHz, PalmOS v. 3.1) (ms)
ECC Benchmarks (163-bit)			
Key Generation	372.4	514	556
Key Expansion[1]	254.8	350	378
Diffie-Hellman Key Agreement	335.6	462	500
ECC-DSA Signature Generation	514.8	713	773
ECC-DSA Signature Verification	1254	1740	1885
RSA Benchmarks(1024-bit)[2]			
Private Encrypt	21734	27808	29628
Public Decrypt (e=3)	598	758	790
Public Decrypt (e=65537)	1482	1860	1966
Public Encrypt	622	798	834

Figure 3: Execution times for RSA and ECC cryptographic primitives

Notes about the benchmarks:

- The RSA benchmark program used the RSA, SHA, big number, and random number modules of the SSLeay cryptographic library that was ported to Palm OS by Ian Goldberg. The particular version of SSLeay that was used for the benchmarks described here was optimized by Ian Goldberg and Nagendra Modadugu, and has been used at the Network Security Research group at Stanford University for security-related research.

- The ECC benchmark program was built using Certicom's Security Builder Release 2.2 for Palm OS.

- CodeWarrior Release 6 for Palm OS was used to compile both the RSA and ECC benchmarks.

- The RSA benchmark program was written by Rob Lambert's team at Certicom. The ECC benchmark program was written by Chris Hawk at Certicom. For the particular operations that were used in this report, the source code of both of the ECC and RSA benchmark programs were code reviewed to ensure that the programs execute those cryptographic operations that they claim to execute in their respective user interfaces. In addition, the source code was independently re-compiled, linked, and executed to generate the timing measurements described in this report.

4. WTLS Handshake Timing Estimates

[1] Certicom's ECC library requires that public keys be expanded into a more efficient representation before they can be operated on. These key expansions are not necessary in an RSA-based handshake, and hence the extra time to execute these operations was also modeled in the benchmarks.

[2] The decryption timing measurements for RSA were measured for both of e=3 and e=65537. It should be noted that e=65537 is more commonly used for most security applications and public decryption operations take longer to execute with e=65537.

The WTLS handshake timing estimates described in this section were derived by adding together the appropriate execution times for the cryptographic primitives from Section 3 involved in the ECC-based and RSA-based handshakes as described in Section 2.

Server-Authenticated Only WTLS Handshake

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	CA Public Key Expansion	254.8
	ECC-DSA Signature Verification	1254
Session Key Establishment	ECC Key Generation (DH Ephemeral Key)	372.4
	Server Public Key Expansion	254.8
	Key Agreement	335.6
TOTAL		**2471.6**

Figure 4: ECC Server-Authenticated Only WTLS Handshake on a 20 MHz Palm VII

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	RSA Signature Verification (Public decrypt, e=3)	598
Session Key Establishment	RSA Encryption (Public encrypt)	622
TOTAL		**1220**

Figure 5 (a): RSA Server-Authenticated Only WTLS Handshake on a 20 MHz Palm VII (e=3)

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	RSA Signature Verification (Public decrypt, e=65537)	1482
Session Key Establishment	RSA Encryption (Public encrypt)	622
TOTAL		**2104**

Figure 5 (b): RSA Server-Authenticated Only WTLS Handshake on a 20 MHz Palm VII (e=65537)

- Using ECC technology to execute the public-key cryptographic operations for a server-authenticated only handshake, the necessary operations take 2.47 seconds. This measurement was determined by using Certicom's Security Builder product for Palm devices, and used an ECC key length of 163 bits.

- Using RSA technology to execute the necessary cryptographic operations for a server-authenticated only handshake, the necessary operations take 1.22 seconds for e=3, and 2.10 seconds for e=65537. This measurement was determined by using an optimized version of Ian Goldberg's port of Eric Young's SSLeay package to the Palm. The measurement used a RSA key length of 1024 bits.

- **The cryptographic execution time for server-authenticated 1024-bit RSA handshakes is up to 2 times as fast as the cryptographic execution time for server-authenticated 163-bit ECC handshakes on the Palm VII.**

Mutual-Authentication WTLS Handshake

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	CA Public Key Expansion	254.8
	ECC-DSA Signature Verification	1254
Session Key Establishment	Server Public Key Expansion	254.8
	Key Agreement	335.6
Client Authentication	ECC-DSA Signature Generation	514.8
TOTAL		**2614**

Figure 6: ECC Mutual-Authentication WTLS Handshake on a 20 MHz Palm VII

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	RSA Signature Verification (Public decrypt, e=3)	598
Session Key Establishment	RSA Encryption (Public encrypt)	622
Client Authentication	RSA Signature Generation (Private encrypt)	21734
TOTAL		**22954**

Figure 7(a): RSA Mutual-Authentication WTLS Handshake on a 20 MHz Palm VII (e=3)

Operation	Cryptographic Primitive(s)	Time Required (ms)
Server Certificate Verification	RSA Signature Verification (Public decrypt, e=65537)	1482
Session Key Establishment	RSA Encryption (Public encrypt)	622
Client Authentication	RSA Signature Generation (Private encrypt)	21734
TOTAL		**23838**

Figure 7(b): RSA Mutual-Authentication WTLS Handshake on a 20 MHz Palm VII (e=65537)

- Using ECC technology to execute the public-key cryptographic operations for a mutually authenticated handshake, the necessary operations take 2.61 seconds. This measurement was determined by using Certicom's Security Builder product for Palm devices, and used an ECC key length of 163 bits.

- Using RSA technology to execute the necessary cryptographic operations for a mutually authenticated handshake, the necessary operations take 22.9 seconds for e=3 and 23.8 seconds for e=65537. This measurement was determined by using an optimized version of Ian Goldberg's port of Eric Young's SSLeay package to the Palm. The measurement used a RSA key length of 1024 bits.

- **The cryptographic execution time for mutually-authenticated 163-bit ECC handshakes is at least 8.64 times as fast as the cryptographic execution time for mutually-authenticated 1024-bit RSA handshakes on the Palm VII.**

5. Notes / Comments

- The benchmarks designed and implemented for the purposes of this paper strictly measure the CPU time required for the execution of cryptographic operations involved in a WTLS handshake. In a real end-to-end WTLS handshake between a Palm VII device and a WAP gateway, network latencies, data transmission, and lower-level protocol handshake times would also be involved in execution of the protocol. (Network latencies and lower-level protocol handshake times should be similar regardless of whether an RSA-based or ECC-based WTLS handshake is measured. However, one would expect that the data transmission times in a RSA-based WTLS handshake should be longer due to longer keys and certificates being transmitted across a wireless network for handshakes of comparable cryptographic strength.)

- For ECC-based implementations of WTLS, the public keys for all trusted CAs can be stored in expanded form to eliminate the need to expand these keys just prior to connection setup. If this is done for server-authenticated only connections, the total time spent executing cryptographic operations would be 2.22 seconds, which is comparable to the 2.10 seconds that would be required for an RSA-based server-authenticated connection where e=65537. Also, by making this optimization, ECC-based implementations of mutually-authenticated WTLS can be made to execute 10.75 times faster than an RSA-based implementation where e=65537.

- 1024-bit RSA-based, mutually-authenticated handshakes will most likely be take too long for most users, and would affect the usability of secure wireless transactions. Shorter key-lengths for RSA-based handshakes could be used to produce a more acceptable user experience, but service providers may not be willing to take the risk or provide certain applications under shorter key lengths. ECC-based handshakes may be the best option for service providers and users that would like mutual authentication.

- This paper reports the results of timing benchmarks on the Palm VII device as a prototypical wireless PDA. As a reference for comparison, listed below are processors and clock cycle speeds for several other popular PDAs in the market. The various handheld devices listed below have a variety of processor types (both CISC and RISC), and a range of click cycle speeds (10 MHz to 190 MHz). RIM devices run at the slowest clock speed (10 MHz) with Intel 386 microprocessors while the Windows CE devices run at the fastest clock speeds with NEC and Intel RISC microprocessors.

PDA	Microprocessor	Speed
Palm, Handspring	Motorola Dragonball	16.6 – 20 MHz
RIM Interactive Pager	Intel 386	10 MHz
Compaq Aero 1530	NEC/VR4111 MIPS RISC	70 MHz
HP Jornada 820	Intel/StrongARM RISC SA-1100	190 MHz
Casio Cassiopeia E-100	NEC/VR4121 MIPS	131 MHz
Psion Revo	ARM 710	36 MHz
Psion Series 5	Digital/Arm 7100	18

Figure 8: Microprocessors and clock speeds of some popular PDAs.

If the benchmarks from Section 3 were run on the various devices in Figure 8, it is reasonable to expect the actual speeds would likely vary proportional to the clock speed of the microprocessor, but the ratios involved would be constant. For example, the ratio of ECC-DSA verification time to signature time should still be approximately 2.4:1. .

Hence, the execution times for server-authenticated RSA handshakes should be approximately twice as fast as for server-authenticated ECC-based handshakes, and the execution time for mutually-authenticate handshakes should be approximately eight times faster using ECC-based handshakes than RSA-based handshakes, independent of the particular microprocessor and its clock cycle speed.

6. Conclusion

To date, the few WAP-browsers that exist for wireless PDAs do not support WTLS. However, WTLS will be instrumental in providing security for WAP services. It is important to understand the cryptographic efficiency of WTLS on these constrained devices as it affects usability of these devices for secure transactions. In studying the WTLS protocol, the cryptographic requirements of the protocol, and the time required to execute the required operations, we have found that 1024-bit RSA-based handshakes can be up to twice as fast as 163-bit ECC-based handshakes for server-authenticated only WTLS connections, and that 163-bit ECC-based handshakes are at least 8 times as fast as 1024-bit RSA-based handshakes for mutually-authenticated WTLS connections.

7. References

[1] WAP Forum, Wireless Application Protocol Specification Version 1.1, 4.30.1998

[2] WAP Forum, Wireless Transport Layer Security Specification Version 1.1, 11.2.1999

[3] AU-Systems WAP Browser Home Page, http://www.wapguide.com/wapguide/browser.html

[4] EZOS EzWAP Browser Page, http://www.ezos.com/

[5] Psion WAP Browser Beta Page, http://wap.psion.com/

[6] Neomar RIM WAP Browser Page, http://www.neomar.com/

[7] Neomar Press Release, http://www.neomar.com/press/00.05.23certicom.html

Acknowledgements

Thanks to Tim Dierks, Rob Lambert and his team, and Chris Hawk for authoring the ECC and RSA benchmarking programs.

Thanks to Nagendra Modadugu for his support in helping prepare the RSA benchmark to compile against his optimized version of the SSLeay library, and for generating some of the preliminary RSA timing measurements that helped validate measurements in this report.