

Comparison Studies between Pre-Shared and Public Key Exchange Mechanisms for Transport Layer Security

Fang-Chun Kuo, Hannes Tschofenig, Fabian Meyer and Xiaoming Fu

Institute for Informatics, University of Goettingen, Germany, Email: {fkuo,fmeyer,fu}@cs.uni-goettingen.de
Corporate Technology, Siemens AG, Munich, Germany, Email: hannes.tschofenig@siemens.com

Abstract—The pre-shared key based mechanisms for Transport Layer Security (TLS) were recently standardized by the IETF to extend the set of ciphersuites by utilizing existing key management infrastructures. The benefit of pre-shared based mechanisms is the avoidance or reduction of the cryptographic operations used in public-key based mechanisms. However, so far there are no performance measurements for pre-shared key based ciphersuites available. In this paper, we present a systematic analysis and performance comparison between the pre-shared key exchange mechanisms and the standard public key exchange mechanisms in TLS. Our performance metrics are processing time and transmitted amount of data for a handshake establishment. Furthermore, the interaction between the overall TLS handshake duration and the network environment is evaluated. The results for different key exchange mechanisms are comparatively studied and the design choices of pre-shared key based key exchange mechanisms have been validated. Experimental results give details about the performance improvement of the pre-shared key based mechanisms compared to the standard public key based mechanisms.

I. INTRODUCTION

In recent years, with the rising threats of data tampering and data interruption, security is becoming an important issue for data communication over the Internet. Secure Socket Layer (SSL) [1] which was later standardized as Transport Layer Security (TLS) [2] is widely used for protecting web traffic and many other application protocols.

The TLS protocol comprises two main components, namely the TLS Handshake protocol, which is responsible for negotiating and establishing secure connections, and the TLS Record protocol, which is responsible for securing the data transmission. The TLS handshake uses certificates and a Public Key Infrastructure (PKI) for mutual authentication and key exchange. However, in some deployment environments, a TLS public-key based handshake may be unnecessary, e.g. the environment suitable for the 3GPP Generic Bootstrapping Architecture [3]. In such cases it might be desired to rely on a secret that is shared in advance between the TLS client and server via out-of-band means. Thus, a set of pre-shared key based ciphersuites for TLS was proposed in [4]. Furthermore, the public-key based handshake process of TLS is part of the bottleneck that significantly degrades the performance [5]. Hence, owing to the reduction of cryptographic operations,

the pre-shared key based mechanisms of TLS are more suitable for performance-constrained environments, e.g. wireless communications, and devices with limited CPU power. So far, there are no performance measurements for pre-shared key based ciphersuites available. Nevertheless, the effect of key exchange mechanism selection on the performance should be analyzed in order to determine the trade-off. Therefore, the objective of this paper is to take a close and critical look at the public key and pre-shared key based mechanisms for TLS handshake protocol with an eye on performance.

The performance of TLS has been extensively evaluated in the literature. [6] and [7] analyzed the architectural impact of SSL usage at TLS web servers. The impact of each individual operation of the TLS protocol has been studied in [7] in the context of Web Servers. In [8] and [5], the performance of a full handshake was analyzed and optimizations for the TLS protocols were also presented. For handheld mobile devices [9] presented a performance analysis to show that the cryptographic operations in security protocols, including SSL, do not significantly impact real-time mobile transactions. Related publications, such as [10] and [11], provided performance evaluations for Wireless Transport Layer Security (WTLS) [12], which is similar to TLS.

Previous attempts to understand TLS performance have focused on the performance of both data transfer and handshake phases in the server side. However, since the data transfer phase of pre-shared key based TLS is the same as that of standard public key based TLS, we examine only the handshake phase in both server and client sides. Firstly, the client and server processing time and the transmitted data amount are measured. Besides the device processing time for a handshake, the overall handshake duration also involves other delays due to message parsing and network latency. The relative overhead due to increase in data volume depends on the type of network deployment. Hence the overall TLS handshake duration is analyzed by considering the client and server processing time as well as the interaction with the network environment. It can be seen in our evaluation results that the pre-shared key ciphersuites perform better than the comparative public key ciphersuites.

The remainder of the paper is organized as follows. In

Section II, an overview of the TLS protocol is briefly presented. More detailed pre-shared key based mechanisms of TLS are described and compared in Section III. Section IV is devoted to the performance analysis of different key exchange mechanisms in TLS. Finally, Section V concludes the paper.

II. AN OVERVIEW OF TLS

A TLS connection is divided into two stages, the *handshake* and *data transfer* phase [2]. The main goal of the TLS protocol is to support security services with confidentiality, integrity and authentication between two communicating applications. To achieve this goal, the TLS protocol utilizes hybridcipher-methods to authenticate the parties and cipher connections as well as data. More specifically, asymmetric techniques, such as RSA [13], are used in the handshake phase for authentication and cipher-key-exchange while symmetric techniques, such as DES [14], are used in the data transfer phase to cipher data and connections as well as check the data integrity. During the establishment process of a TLS session¹, the handshake phase authenticates the server and the client (optional) and establishes the necessary keys to protect the data transmission. Once the handshake is successfully completed, the session enters the data transfer phase, thus the application data can be exchanged over a secure connection.

Figure 1 shows the full handshake message flow required to establish a new session. However, the full TLS handshake sequence may vary, depending on whether the public or pre-shared key exchange is used. The TLS Handshake protocol is responsible for negotiating a session. There are three purposes for the TLS handshake protocol. First, the client and the server have to agree on a set of algorithms for data protection by sending **ClientHello** and **ServerHello** messages as shown in Figure 1. The second purpose is to establish the keys to be used in those algorithms. The third purpose is to authenticate the server and optionally to authenticate the client. Key establishment and authentication are achieved by transmitting **ServerKeyExchange** and **ClientKeyExchange** messages. Once a side has sent its **Finished** message as well as received and verified the **Finished** message from its peer, it may begin to transmit or receive application data over the secure connection.

III. PRE-SHARED KEY EXCHANGE MECHANISMS

Pre-shared key exchange mechanisms for TLS protocol are proposed in [4] for the sake of supporting authentication based on pre-shared symmetric keys. In [4] three key exchange suites using pre-shared key exchange mechanism are discussed. The first suite uses only pre-shared key (PSK) mechanism, namely Plain PSK. The second suite is DHE.PSK which uses a pre-shared key (PSK) to authenticate a DH key exchange. Finally, the third suite is RSA.PSK which combines public key based authentication of the server using RSA and certificate with mutual authentication using a PSK.

¹A TLS session is negotiated on a handshake. Each session is identified by a session ID allocated by the server. The items, e.g. session ID, ciphers and master secret, negotiated during the session are used for setting up secure connections.

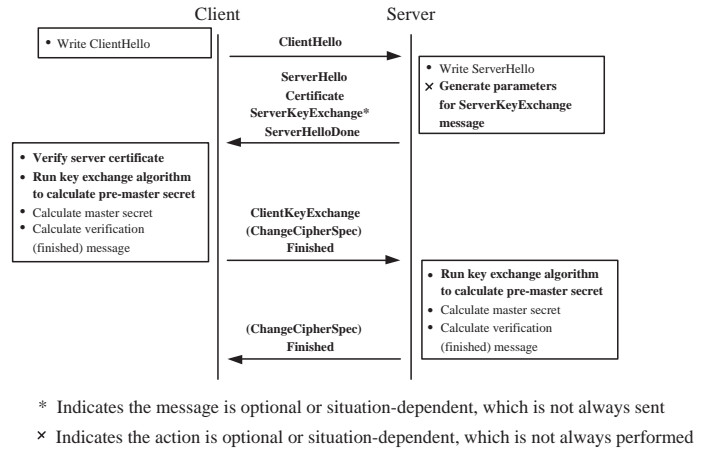


Fig. 1. Full TLS handshake with server authentication [11]

A. Plain PSK

The client should indicate which key to use by transmitting a “PSK identity” to the server because both clients and servers may have pre-shared keys with several different parties. As shown in Figure 2, the “PSK identity” is included in the **ClientKeyExchange** message and transmitted to the server. After the negotiation for “PSK identity” is done, the client and the server can generate their pre-master secrets with the pre-shared key. As normal TLS process, the **Finished** message is computed for the authentication of the TLS handshake. Furthermore, the plain PSK key exchange mechanism uses only symmetric key algorithms and consequently there is no burden caused by public key operations. Hence, the handshake time of plain PSK is even faster.

Note that PSK, DHE.PSK and RSA.PSK share the same general structure for the pre-master secret which is generated by including the “pre-shared key” and “other secret”. In plain PSK key exchange mechanism, “other secret” is ZEROS while “other secret” comes from the DH or RSA exchange in DH.PSK and RSA.PSK, respectively.

B. DHE.PSK

The mechanism of DHE.PSK is similar to that of DHE.DSS using server authentication. However, DHE.PSK utilizes the pre-shared key (PSK) to authenticate a DH exchange instead of a DSS signature. Thus no client and server certificates are transmitted and no digital signatures are operated during the TLS session when using DHE.PSK. DH uses the **ServerKeyExchange** and **ClientKeyExchange** messages to transmit the DH parameters. As depicted in Figure 3, the server generates the DH server keys and sends the keys in the **ServerKeyExchange** message. Then the client generates the DH client keys and includes the keys as well as the “PSK identity” in the **ClientKeyExchange** message. To generate the pre-master secret, first, the normal DH computation is performed and then the pre-master secret is generated with the computed DH value and the pre-shared key.

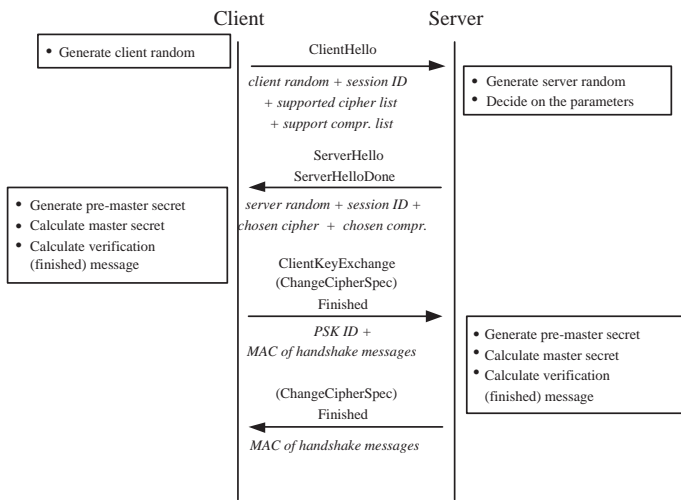


Fig. 2. Full handshake for Plain PSK.

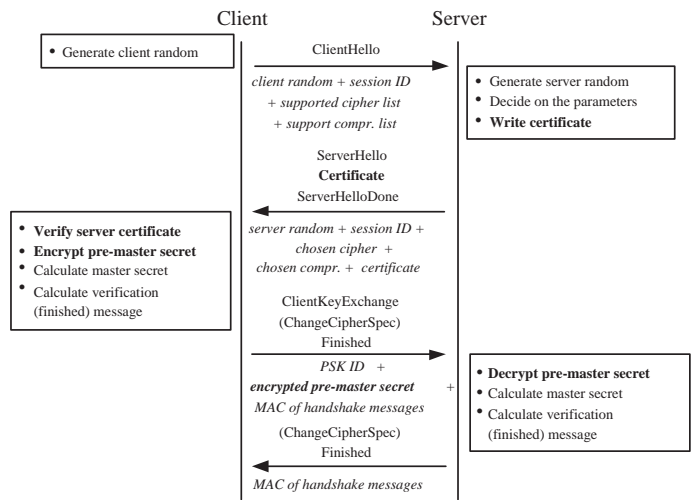


Fig. 4. Full handshake for RSA_PSK.

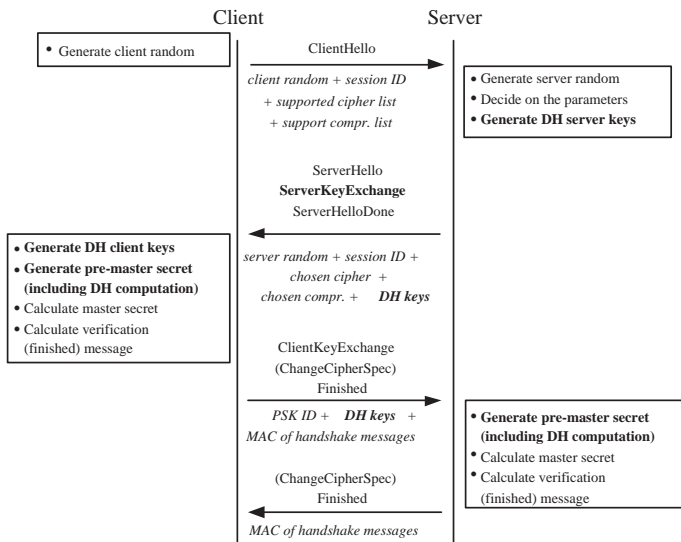


Fig. 3. Full handshake for DHE_PSK.

C. RSA_PSK

The RSA_PSK key exchange mechanism uses RSA and a certificate to authenticate the server and also achieves the mutual authentication by using PSK. As shown in Figure 4, RSA_PSK is similar to RSA using server authentication. The server has to send a **Certificate** including the server's public key to the client. The client encrypts the pre-master secret using server's public key from the received **Certificate** message. Then the server decrypts the pre-master secret using its private key. The difference between RSA_PSK and RSA is in the generation of the pre-master secret. In RSA_PSK, the pre-master secret is generated by including the RSA random value and the pre-shared key. Since the server has received the "PSK identity", the server can compare the pre-shared key it has with the client's pre-shared key extracted from the decrypted pre-master secret. The client is authenticated

by the server if both match. Therefore, RSA_PSK uses the pre-shared key and server certificate for mutual authentication while RSA with mutual authentication uses both server and client certificates. In other words, RSA_PSK can save the overhead due to transmission and verification of the client certificate but still achieve mutual authentication.

D. Security Considerations

Since these pre-shared keys are shared in advance to avoid public key operations, it is useful for performance-constrained environments. Furthermore, in some cases a fully general public-key-based handshake is unnecessary, e.g. 3GPP cellular mechanisms, since the parties already have a mechanism for setting up a shared secret key. However, special care should be taken for protecting the pre-shared secrets. Use of a fixed pre-shared secret of limited entropy (e.g. a short secret that is chosen by a human) allows the attacker to connect to the server and try different keys to recover the secret by off-line or on-line attack. The limitation due to the configuration of the pre-shared secret reduces the security of the deployment. It is recommended in [4] that the configuration of the pre-shared secret should provide a functionality for generating a new random pre-shared secret. Moreover, the static pre-shared secret key is somehow compromised, and thus an attacker can decrypt old conversations. Nevertheless, among the ciphers based on PSK key exchange mechanism, only DHE_PSK can provide Perfect Forward Secrecy (PFS)² to ensure that a fresh DH private key is generated for each handshake.

²In the DHE (*ephemeral* DH) key establishment algorithm, the sever and client generate temporary private keys only for each handshake. DHE provides *Perfect Forward Secrecy* (PFS) because the attacker cannot recover any old data once the parties are done communicating and delete their temporary private keys. If one of the key was static, an attacker might be able to break into the machine with the key and retrieve it.

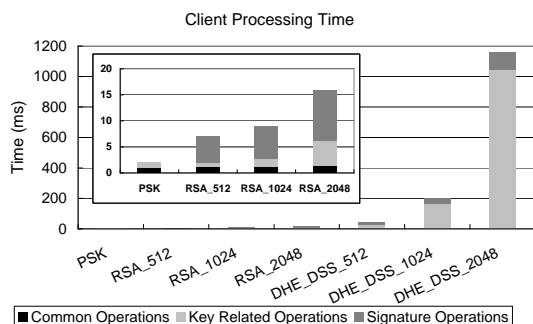


Fig. 5. Client Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE.DSS using server authentication.

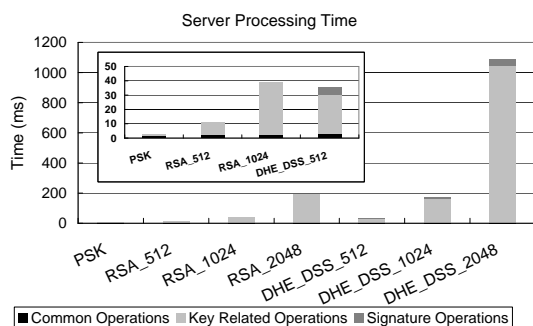


Fig. 6. Server Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE.DSS using server authentication.

IV. PERFORMANCE EVALUATION

A. Performance Metrics

Previous studies of TLS performance focused on the performance of the data transfer and handshake phase, especially in the server side. Since there is no difference in the data transfer phase between pre-shared and public key based mechanisms, we examine only the handshake phase in both server and client sides. However, besides server and client processing time, other delays due to message parsing and network latency have to be taken into account in the duration for a TLS handshake. Therefore the transmitted data amount of a handshake for different key exchange mechanisms is also measured to analyze the interaction between the overall TLS handshake duration and the network environment. In summary, there are three metrics for performance comparison in the TLS handshake phase:

- 1) Client and Server processing time
- 2) The amount of the transmitted data
- 3) Handshake duration

The client and server machines are equipped with a VIA Samuel 2 processor running at 533 Mhz with 256 MB of RAM and 20 GB HDD. The used PSK-TLS implementation is to be released as part of OpenSSL (an open source toolkit for SSL/TLS).

B. Client and Server Performance

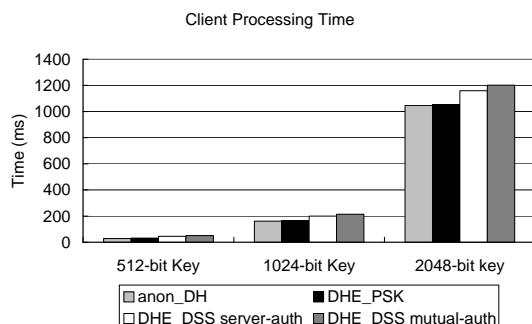


Fig. 7. Client Processing Time in the handshake protocol when using anonymous DH, DHE.PSK, DHE.DSS using server authentication and the DHE.DSS using mutual authentication.

1) *Plain PSK*: Figure 5 and Figure 6 show the client and the server handshake processing time of the plain PSK, RSA using server authentication and DHE.DSS using server authentication. There are three different types of processing time in the device processing time, namely *key related operations*, *signature operations* and *common operations*. The detailed location of performance observation points can be found in [15]. As seen in Figure 5 and Figure 6, common operation time of the plain PSK is similar to that of RSA and DHE.DSS and the key related operation time of the plain PSK is negligibly small when compared to RSA and DHE.DSS. This implies that the plain PSK significantly reduces the client and the server processing time due to the lack of public key cryptographic operations.

2) *DHE_PSK*: Figure 7 and Figure 8 show the client and the server processing performance of DHE_PSK compared to that of anonymous DH, DHE.DSS using server authentication and DHE.DSS using mutual authentication. The mechanism of anonymous DH is similar to that of DHE.DSS but without the signature process while the mechanism of DHE_PSK is also similar to that of DHE.DSS but uses pre-shared keys to authenticate the server and the client instead of the signature process. Hence the processing time of DHE_PSK is just slightly greater than that of anonymous DH. Furthermore, compared to DHE.DSS, the performance improvement of DHE_PSK in device processing time is slight because the DH key computation process consumes much more time than the DSS signature process as seen in Figure 5 and Figure 6.

C. The Amount of transmitted Handshake Data

The transmitted handshake data amounts of different key exchange mechanisms are analyzed in Table I. The plain PSK mechanism largely reduces the transmitted data amount because in plain PSK only the information of the pre-shared key is exchanged during the handshake process but no certificate is transmitted. Furthermore, anonymous DH and DHE_PSK need less data to be exchanged than other public key exchange mechanisms that use server authentication or mutual authentication since anonymous DH and DHE_PSK do not need to transmit any certificates during the handshake. Following this philosophy, it can be inferred that mechanisms using

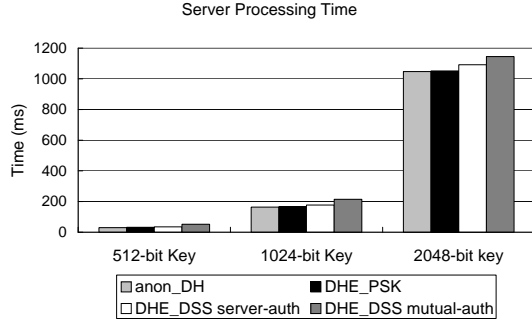


Fig. 8. Server Processing Time in the handshake protocol when using anonymous DH, DHE.PSK, DHE.DSS using server authentication and the DHE.DSS using mutual authentication.

TABLE I

THE AMOUNT OF TRANSMITTED DATA DURING A TLS HANDSHAKE.

Mechanism	Data Amount (bytes)		
	512-bit key	1024-bit key	2048-bit key
Plain PSK	586		
DH.anon	719	911	1295
DHE.PSK	791	983	1367
RSA	1242	1439	1861
DHE.DSS	1558	1950	2821
RSA mutual auth	1994	2388	3298
DHE.DSS mutual auth	2417	3009	4260

mutual authentication need more data to be transmitted than mechanisms using only server authentication. In conclusion, the transmitted data amount of plain PSK is the smallest due to the lack of the transmission of the public key and certificates. On the contrary, DHE.DSS using mutual authentication has the largest data amount.

D. Handshake Duration Analysis

After analyzing the device processing time and the amount of transmitted handshake data, we are going to analyze the impact of the different components on the overall duration of a handshake. For the simplicity, we assume that the TLS handshake duration is the sum of the client and server processing time and data transmission time. While the data transmission time is simply calculated as

$$T_{\text{Data Transmission}} = \frac{\text{Transmitted handshake data amount}}{\text{Network Throughput}} \quad (1)$$

1) *Plain PSK*: Figure 9 depicts the handshake duration of plain PSK, RSA and DHE.DSS. RSA and DHE.DSS use only server authentication. As expected, the handshake duration increases when the network throughput decreases since the data transmission time increases rapidly when network throughput decreases. However, the data transmission time depends not only on network throughput but also on the transmitted data amount. Because the transmitted data amount of plain PSK is much less than that of RSA and DHE.DSS as illustrated in Table I, the handshake duration of plain PSK increases more slowly when network throughput decreases as shown in Figure 9. Accordingly, plain PSK is suitable for constrained channel condition, e.g. wireless communication.

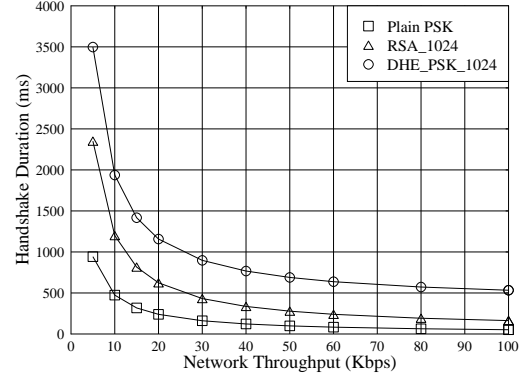
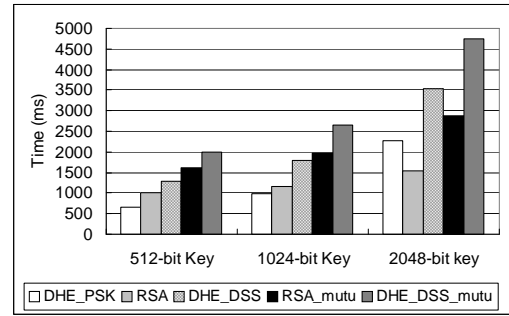
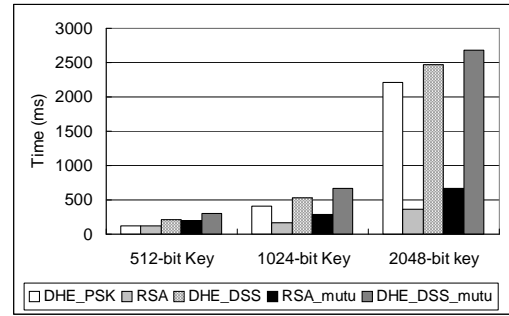


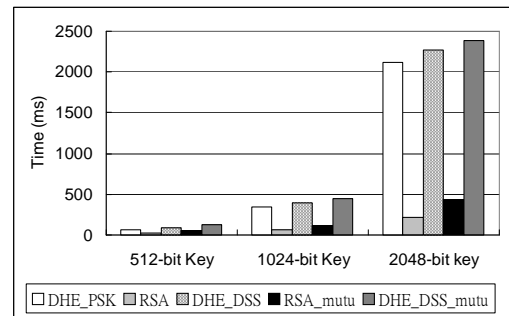
Fig. 9. Handshake Duration of Plain PSK, RSA using server authentication, and DHE.DSS using server authentication. The key size for RSA and DHE.DSS is 1024-bit



(a) Throughput = 10 Kbps



(b) Throughput = 100 Kbps



(c) Throughput = 1 Mbps

Fig. 10. Handshake Duration of DHE.PSK, DHE.DSS using server authentication, DHE.DSS using mutual authentication, RSA using server authentication and RSA using mutual authentication at 10k, 100k and 1M bps network throughput.

2) *DHE_PSK*: In Figure 10, the handshake duration of *DHE_PSK* is compared with *DHE_DSS* and *RSA* using server and mutual authentication under different network throughput values, 10 K, 100 K and 1 M bps. As depicted in Figure 10, *DHE_PSK* always performs better than *DHE_DSS* using server or mutual authentication because *DHE_PSK* needs less device processing time and a smaller amount of transmitted data. Nevertheless, when compared to *RSA*, the handshake duration of *DHE_PSK* depends on key size and network condition. When using small key sizes with low network throughput, *DHE_PSK* performs better than *RSA* in terms of handshake duration as shown in Figure 10(a). On the contrary, the handshake duration of *DHE_PSK* is much longer than that of *RSA* in high network throughput environments as shown in Figures 10(b) - 10(c).

Note that the performance improvement achieved by *RSA_PSK* over *RSA* using mutual authentication should be similar to the improvement achieved by *DHE_PSK* over *DHE_DSS* using server authentication since both *RSA_PSK* and *DHE_PSK* use pre-shared keys to authenticate each other instead of a digital signature. Therefore, the performance of *RSA_PSK* can be estimated by comparing *DHE_PSK* and *DHE_DSS*. However, the validity needs further in-depth research on *RSA_PSK* and *RSA*, which will be our future work.

V. CONCLUSIONS

We have presented a systematic analysis of performance and a comparison between the pre-shared and public key exchange mechanisms for TLS. We have shown that owing to the smallest transmitted data amount as well as the shortest device processing time, plain PSK performs better than any other public key based mechanisms and the handshake duration of plain PSK increases slowly when network throughput decreases. For *DHE_PSK*, the handshake duration is shorter than that of *DHE_DSS* using server or mutual authentication because *DHE_PSK* needs less device processing time and a smaller amount of transmitted data. However, when compared to *RSA*, *DHE_PSK* performs better than *RSA* only when using small key sizes and having low network throughput. Although *DHE_PSK* may perform worse than *RSA* when using large key sizes or high network throughput, *DHE_PSK* provides Perfect Forward Secrecy (PFS) to ensure a more secure communication among the pre-shared key based ciphers.

REFERENCES

- [1] A. Frier, P. Karlton, and P. Kocher, "The Secure Socket Layer," Netscape Communications Corp., Tech. Rep., November 1996.
- [2] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," Internet Engineering Task Force, RFC 2246, January 1999.
- [3] "Generic Authentication Architecture (GAA); Generic bootstrapping architecture, 3GPP TS 33.220 V7.2.0," 3rd Generation Partnership Project Technical Specification, Dec. 2005, <http://www.3gpp.org/ftp/Specs/html-info/33220.htm>.
- [4] P. Eronen and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)," Internet Engineering Task Force, RFC 4279, dec 2005.
- [5] G. Apostolopoulos, V. G. J. Peris, and D. Saha, "Transport Layer Security: How Much Does It Really Cost?" in *INFOCOM*, 1999, pp. 717-725.

- [6] K. Kant, R. Lyer, and P. Mohapatra, "Architectural impact of secure socket layer on internet servers." in *International Conference on Computer Design (ICCD)*, 2000.
- [7] C. Coarfa, P. Karlton, and D. Wallach, "Performance analysis of TLS Web Server," in *Proceedings of NDSS '02*, 2002.
- [8] A. Goldberg, R. Buff, and A. Schmitt, "Secure Web Server Performance Dramatically Improved By Caching SSL Session Keys," in *Workshop on Internet Server Performance*, June 1998.
- [9] P. G. Argyroudis, R. Verma, H. Tewari, and D. O'Mahony, "Performance Analysis of Cryptographic Protocols on Handheld Devices." in *NCA*, 2004, pp. 169-174.
- [10] I. Herwono and I. Liebhardt, "Performance of WTLS and Its Impact on an M-commerce Transaction." in *ICICS*, 2001, pp. 167-171.
- [11] A. Levi and E. Savas, "Performance Evaluation of Public-Key Cryptosystem Operations in WTLS Protocol," in *The 8th IEEE Symposium on Computers and Communications - ISCC 2003*, July 2003, pp. 1245-1250.
- [12] WAP Forum, "Wireless Transport Layer Security Specification," Wireless Application Protocol Forum Ltd., Tech. Rep., April 2001, <http://www1.wapforum.org/tech/documents/WAP-261WTLS-20010406-a.pdf>.
- [13] R. L. Rivest, A. Shamir, and L. M. Adleman, "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science, Technical Report, 1979.
- [14] National Institute of Standards and Technology (NIST), "Data Encryption Standard," U.S. Department of Commerce, December 1993.
- [15] F.-C. Kuo, H. Tschofenig, F. Meyer, and X. Fu, "Comparison Studies between Pre-Shared Key and Public Key Exchange Mechanisms for Transport Layer Security (TLS)," Institute for Informatics, University of Goettingen, Technical Report IFI-TB-2006-01, January 2006.