



Georg-August-Universität
Göttingen
Institut für Informatik

ISSN 1611–1044
Nummer IFI–TB–2006–001

Technischer Bericht

Comparison Studies between Pre-Shared key and Public Key Exchange Mechanisms for Transport Layer Security (TLS)

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

**Technische Berichte
des Instituts für Informatik
an der Georg-August-Universität Göttingen**

January 2006

Georg-August-Universität Göttingen
Institut für Informatik

Lotzestraße 16-18
37083 Göttingen
Germany

Tel. +49 (5 51) 39-1 44 14

Fax +49 (5 51) 39-1 44 15

Email office@informatik.uni-goettingen.de

WWW www.ifi.informatik.uni-goettingen.de

UNIVERSITY OF GOETTINGEN

ABSTRACT

INSTITUTE OF INFORMATICS

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

The public-key based handshake process of TLS is part of bottleneck that significantly degrades the performance. The pre-shared key based mechanisms for Transport Layer Security were recently standardized by the IETF to extend the existing set of ciphersuites by utilizing existing key management infrastructures. The benefit of the pre-shared based mechanisms is the avoidance or reduction of the cryptographic operations used in public-key based mechanisms. Our performance metrics are the processing time in slow processor machines as well as fast processor machines and the transmitted amount of data for a handshake establishment. So far, there are no performance measurements for pre-shared key based ciphersuites available. In this report, we present a systematic analysis and performance comparison between the pre-shared key exchange mechanisms and the standard public key exchange mechanisms in TLS. Furthermore, the interaction of 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 mechanism.

Contents

Abstract	i
List of Figures	iv
List of Tables	v
1 Introduction	1
2 The Overview of Transport Layer Security	4
2.1 TLS Handshake Protocol	4
2.1.1 Full Handshake with Server Authentication	5
2.1.2 Session Resumption Handshake	6
2.2 TLS Record Protocol	7
3 Key Exchange Mechanisms in TLS	9
3.1 Public Key Exchange based Mechanism	9
3.1.1 RSA	9
3.1.2 DHE_DSS	10
3.2 Pre-Shared Key Exchange based Mechanism	12
3.2.1 Plain PSK	13
3.2.2 DHE_PSK	14
3.2.3 RSA_PSK	14
3.3 Functional Comparison of Public and Pre-Shared Key Exchange Mechanisms in TLS	15
4 Performance Evaluation	17
4.1 Performance Metrics	17
4.2 Experimental Setup	17
4.3 Client and Server Performance	18
4.3.1 Plain PSK	18
4.3.2 DHE_PSK	19
4.4 The Amount of transmitted Handshake Data	20
4.5 Handshake Duration Analysis	21
4.5.1 Plain PSK	21
4.5.2 DHE_PSK	22

5 Conclusions

24

Bibliography

25

List of Figures

2.1	Full TLS handshake with server authentication [1]	6
2.2	TLS resumption handshake [2]	7
2.3	TLS data fragmentation process [3]	8
3.1	Full handshake for RSA using server authentication	10
3.2	Full handshake for RSA using mutual authentication [3]	11
3.3	Full handshake for DHE_DSS using server authentication	12
3.4	Full handshake for DHE_DSS using mutual authentication [3]	13
3.5	Full handshake for PSK	14
3.6	Full handshake for DHE_PSK	15
3.7	Full handshake of RSA_PSK mode	16
4.1	Client Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE_DSS using server authentication.	19
4.2	Server Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE_DSS using server authentication.	19
4.3	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.	20
4.4	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.	20
4.5	Handshake Duration of Plain PSK, RSA using server authentication, and DHE_DSS using server authentication in two scenarios, fast server to fast client and slow server to slow client. The key size for RSA and DHE_DSS is 1024-bit	22
4.6	Handshake Duration of DHE_PSK, DHE_DSS using server authentication, DHE_DSS using mutual authentication, RSA using server authentication and RSA using mutual authentication in fast server to fast client scenario at 10k, 100k and 1M network throughput.	23
4.7	Handshake Duration of DHE_PSK, DHE_DSS using server authentication, DHE_DSS using mutual authentication, RSA using server authentication and RSA using mutual authentication in slow server to slow client scenario at 10k, 100k and 1M bps network throughput.	23

List of Tables

4.1	The amount of the transmitted data during a TLS handshake. . . .	21
-----	------------------------------------------------------------------	----

Chapter 1

Introduction

The security is an important issue for data communication over Internet. In recent years, with the rising threats of data tampering and data interruption, security is becoming an important issue for data communication over Internet. The Secure Socket Layer (SSL) [4], the latest version of which is also known as Transport Layer Security (TLS) [5], is by far the dominant approach for securing network traffic [3]. Today SSL/TLS is widely used for protecting web traffic and e-mail protocols such as IMAP and POP. One of the reasons that SSL/TLS has outgrown other transport and application layer security protocol such as SSH [6], SET [7], and S/MIME [8] is that it provides a secure, transparent channel. In other words, it is easy to provide security for an application protocol by inserting TLS between the application and transport layer. There are many examples of applications such as TELNET and FTP running transparently over TLS.

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 not be necessary, e.g. the environment suitable for the 3GPP Generic Bootstrapping Architecture [9]. In such cases it might be desired to rely on a shared secret that is shared in advance between the TLS client and server via some out-of-band means. Thus, a set of pre-shared key based ciphersuites for TLS was proposed in [10]. Furthermore, the use of mobile computing devices, e.g. handhelds, palmtops, mobile phones, has

increased over the years, particularly during the last decade. Typically, the computational constraint of mobile devices is still an obstacle. Designers are forced to use highly efficient protocols for the sake of ensuring the computational complexity of security algorithms as low as possible. However, the public-key based handshake process of TLS is part of the bottleneck that significantly degrades the performance [2]. For a normal transaction (10-15 Kbytes), the TLS handshake consumes most of the CPU time. It has also been observed in [2] that the public key exchange operations in TLS Handshake Protocol consume a significant amount of server resources. Hence, owing to the reduction of the cryptographic operations, the pre-shared key based mechanisms of TLS are more suitable for performance-constrained environments, e.g. wireless communications, and limited CPU power devices. So far, there is no detailed performance measurement comparing the different key exchange suites of TLS, especially for pre-shared key based key exchange suites. However, the effect of key exchange mechanism selection on the performance should be analyzed for the sake of seeing the trade-off. The objective of this report is to take a close and critical look at the public and pre-shared key based ciphersuites for TLS handshake protocol with an eye on performance.

The performance of TLS has been extensively evaluated in the existing literature. For the TLS Web Server performance evaluation, [11] analyzed the performance and architectural impact of SSL on the servers in terms of various parameters such as throughput, utilization, cache sizes and cache miss ratios. The impact of each individual operation of TLS protocol has been studied in [12] in the context of Web Servers and it has been showed that key exchange is the slowest operation in TLS protocol. In [13] and [2], the impacts of full handshake in connection establishment were discussed and optimization methods of the TLS protocols were also presented. For the application of handheld mobile devices [14] presented a performance analysis to show that the cryptographic operations in security protocols, SSL, IPsec and S/MIME, do not significantly impact real-time mobile transactions hence it is feasible to use strong cryptographic protocols on handheld mobile devices. Other related work is the performance evaluation for Wireless Transport Layer Security (WTLS) [15], which is a similar protocol to TLS. In [16], Herwono *et al.* have given performance evaluation of WTLS protocol to measure handshake timing versus network throughput. In [1], Levi and Savas proposed a performance measurement for WTLS protocol to analyze the process time, the amount of data produced and transmitted and the response time.

Previous attempts to understand TLS performance have focused on the performance of both data transfer and handshake phases in the server sides. However, since the data transfer phase of the pre-shared key based TLS is the same as that of the 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 message 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 article is organized as follows. In Chapter 2, an overview of the TLS protocol is briefly presented. More detailed pre-shared key based mechanisms and public key based mechanisms of TLS are described and compared in Chapter 3. Chapter 4 is devoted to the performance analysis of different key exchange mechanisms in TLS. Finally, the conclusions are summarized in Chapter 5.

Chapter 2

The Overview of Transport Layer Security

A TLS connection is divided into two stages, the *handshake* and *data transfer* phase [5]. The main goal of the TLS protocol is to support the security services with confidentiality, integrity and authentication between two communicating applications. To achieve the goal, the TLS protocol utilizes hybridcipher-methods to authenticate the parties and cipher the connection as well as the data. More specifically, asymmetric techniques, such as RSA [17], are used in the handshake phase for authentication and cipher-key-exchange while symmetric techniques, such as DES [18], are used in the data transfer phase to cipher data and connection as well as check 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 goes into the data transfer phase, thus the application data can be exchanged over a secure connection.

2.1 TLS Handshake Protocol

The TLS Handshake protocol is responsible for negotiating a session. There are three purposes of TLS handshake protocol. First, the client and the server have

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.

to agree on a set of algorithms to protect the data. The second purpose is the establishment of keys used in those algorithms. The third purpose is to authenticate the server and optionally to authenticate the client. The TLS Handshake protocol provides a variety of key exchange mechanisms for the sake of key establishment and authentication. The goal of the key exchange process is to generate a *pre-master secret* which is known by communicating parties and protect it from the attackers. Then this pre-master secret is used to create the *master secret*. The master secret is required to generate the **CertificateVerify** and **Finished** messages, encryption key, and MAC secrets.

2.1.1 Full Handshake with Server Authentication

Figure 2.1 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. Moreover, the adoption of the server-authentication, mutual-authentication or anonymous handshake also affect the TLS handshake steps. The detailed description of different key exchange mechanisms will be discussed in the Chapter 3. Here we only provide the general full handshake process with server authentication.

As depicted in Figure 2.1, the client sends a **ClientHello** message which includes a list of ciphers supported by the client and a random number used as input to the key generation process. In response to the **ClientHello** message, the server replies a **ServerHello** message which includes the chosen cipher and a random number, followed by a **Certificate** message that contains server's public key. Note that in some key exchange mechanisms, e.g. DHE.DSS and DHE.RSA, the **ServerKeyExchange** message is sent by the server because the server **Certificate** message doesn't contain enough data to allow exchanging a pre-master secret. Next, the client verifies the server's certificate and extracts server's public key. Then the client runs the key exchange mechanism to generate the pre-master secret and sends out the **ClientKeyExchange** message to the server. After receiving the **ClientKeyExchange** message, the server also runs the key exchange mechanism to generate the pre-master secret. The processes of the certificate verification and the pre-master secret generation are key-exchange-mechanism-dependent, which will be discussed further in Chapter 3. Now both parties have

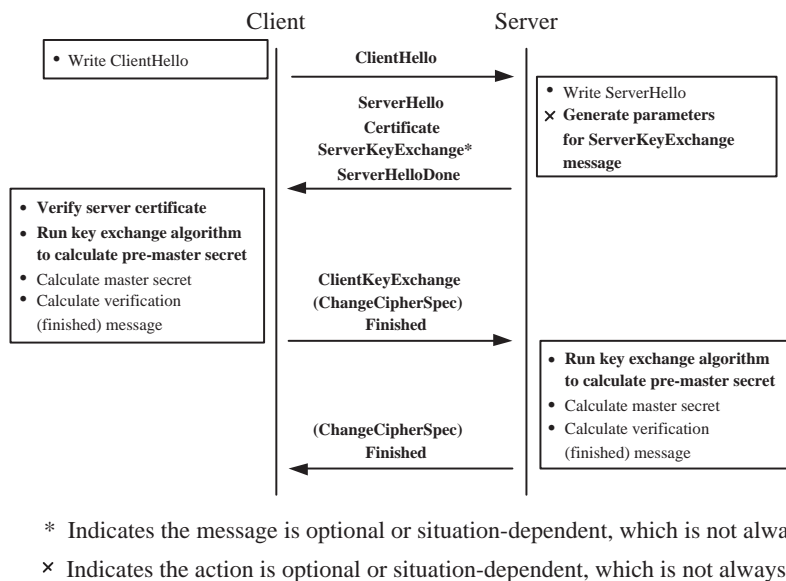


FIGURE 2.1: Full TLS handshake with server authentication [1]

their pre-master secrets which are used to generate master secrets. The master secret is then used to generate symmetric keys for encryption and message authentication. Afterward, the client and the server independently generate the **Finished** messages. The recipients of **Finished** message must verify the content. Once a side has sent its **Finished** message and received and verified the **Finished** message from its peer, it may begin to transmit or receive application data over the connection.

2.1.2 Session Resumption Handshake

When the initial full handshake is finished, both the client and the server store the session information, the keying material and negotiated ciphers in the cache, so that they can get those information later if the session is resumed. Figure 2.2 shows the message flow required to resume a TLS session. Since the session can be identified by the session ID, the client can simply indicate the old or existing session ID it wants to resume by sending the **ClientHello** message. If the server is willing to resume the session, it responds the same session ID by sending the **ServerHello** message. Then the server skips the rest of the handshake and goes to the key generation step directly. The new key is generated by using cached master secret and the new client and server random values for the secure channel, hence they are different from the old keys in the last session. Furthermore, for

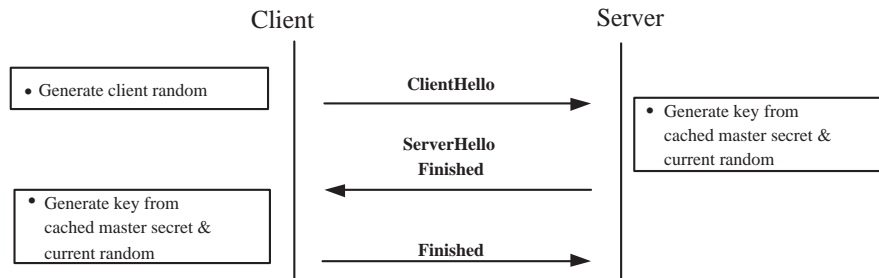


FIGURE 2.2: TLS resumption handshake [2]

the same pair of parties, they can set up the multiple secure links by using single session state [2], which can significantly reduce the latency and the processing involved in the secure channel establishment.

2.2 TLS Record Protocol

The TLS Record protocol is used for encapsulation of various high level protocol , including the TLS Handshake Protocol. The TLS Record protocol provides two basic security service, privacy and message integrity. As mentioned above, TLS Record protocol uses symmetric cryptography, e.g. DES, to encrypt data for privacy service and applies a key message authentication digest, e.g. MD5 [19], to ensure message integrity. As shown in Figure 2.3, in TLS Record Protocol the data stream is divided into a series of *fragments*. For the sake of data integrity protection, a MAC² [20] is calculated and then attached to the data fragment. The concatenated data and the MAC are then encrypted. Finally a header is attached to the encrypted payload as a *record*. Records are what is actually transmitted over secured TLS channel.

²A Message Authentication Code (MAC) is a short piece of information (know as a authentication tag) used to authenticate a message together with a secret key. MACs differ from digital signatures, as MACs are both claculated and verified using the same key, so that they can only be verified by the intended recipient.

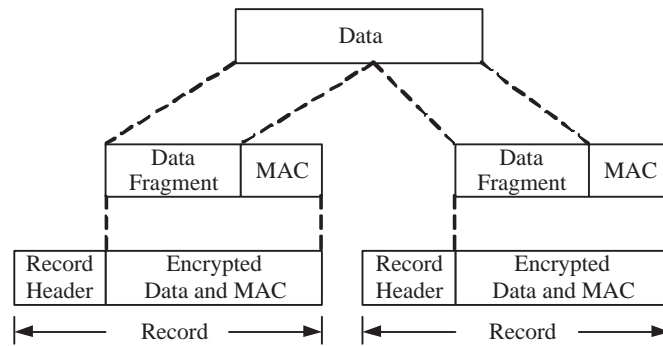


FIGURE 2.3: TLS data fragmentation process [3]

Chapter 3

Key Exchange Mechanisms in TLS

3.1 Public Key Exchange based Mechanism

In the public key based mechanism, the key exchange process can be divided into two parts, *key establishment* and *digital signature*. There are two key establishment algorithms used in standard public key based TLS, namely RSA and Diffie-Hellman (DH) [21]. And RSA and DSS [22] are used as digital signature algorithms in public key exchange TLS. Note that unlike RSA, which can be used for either key establishment or signature, DH can only be used for key agreement and DSS can only be used for signature. However, TLS supports a variety of key exchange suites using public key exchange mechanism. Here we only focus on two most commonly used mechanisms, RSA and DHE_DSS.

3.1.1 RSA

In RSA ciphersuite, RSA is used for key establishment as well as digital signature. For the key establishment, RSA can be used as a *key transport* algorithm, where the sender generates a session key and encrypts it using receiver's public key for the receiver. The receiver then decrypts the message and gets the session key. Using RSA for digital signature is similar to using RSA for key establishment, except that the role of the public and private keys is reversed. The sender calculates a

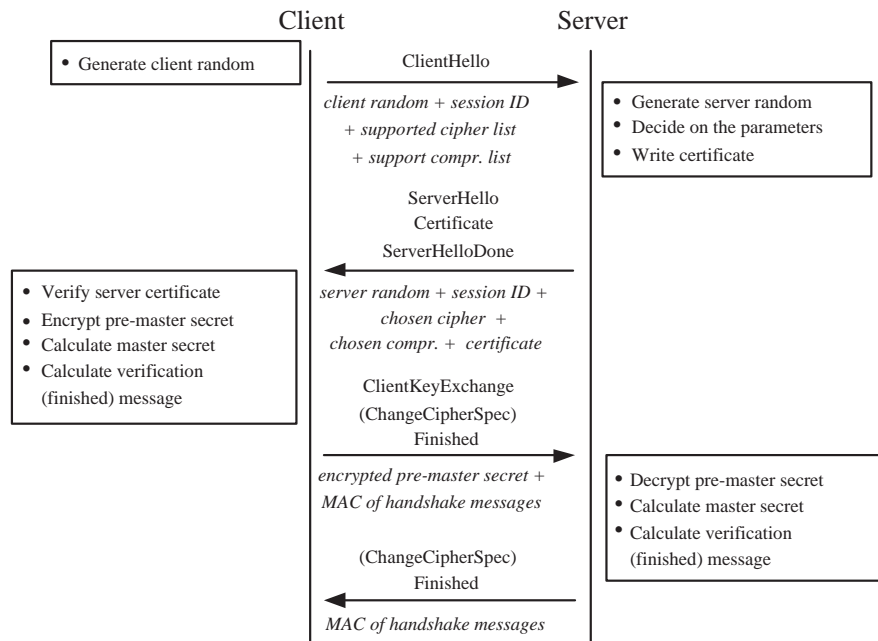


FIGURE 3.1: Full handshake for RSA using server authentication

message digest and encrypts it using sender's private key for the sake of signing. For verification, the receiver decrypts the digest and compares it to the message digest it has independently computed on the message. Then the signature will be announced valid if the comparison matches.

Figure 3.1 shows the full handshake flow of RSA with server-authentication. The server sends the **Certificate** which includes server's public key. The client verifies the **Certificate** with RSA. Then the client generates a random value as pre-master secret and encrypts the pre-master secret with server's public key which is extracted from **Certificate**. After receiving the encrypted pre-master secret included in **ClientKeyExchange** message, the server decrypts it with its private key to recover the pre-master key. Furthermore, the Figure 3.2 shows the RSA with mutual authentication. For achieving client authentication, the server sends the **CertificateRequest** message and the client generates certificate and the RSA signature (included in **CertificateVerify**) for the server to verify them.

3.1.2 DHE_DSS

In DHE_DSS, DH is used for key establishment while DSS is used for digital signature. Diffie-Hellman (DH) is a *key agreement* algorithm rather than a key

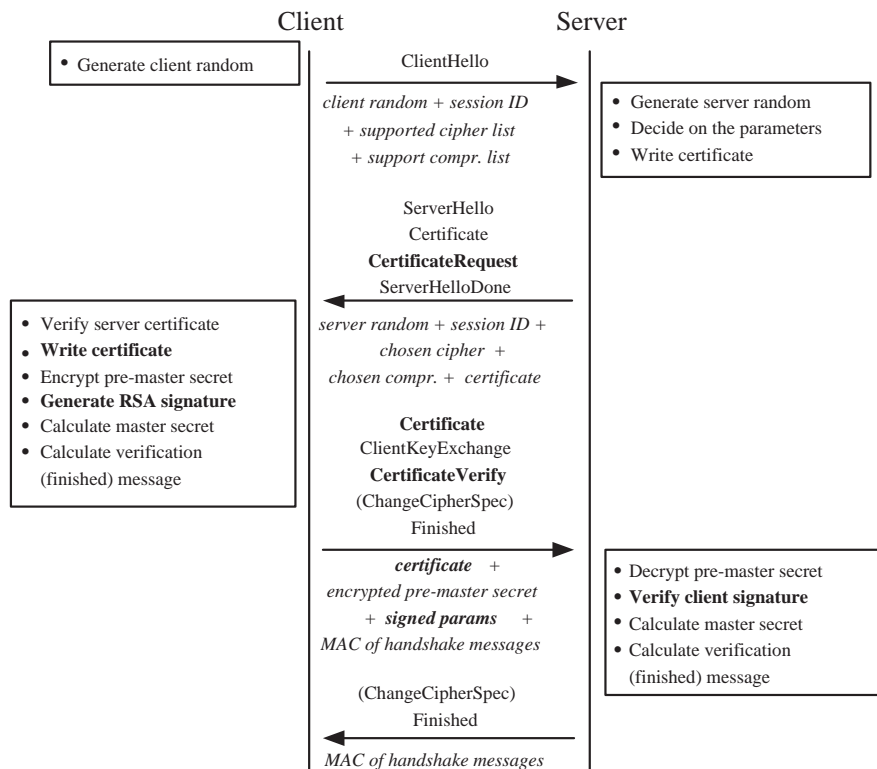


FIGURE 3.2: Full handshake for RSA using mutual authentication [3]

transport algorithm, the sender and receiver collectively generate a key that is private to them. The sender and the receiver each have key pairs. To generate the agreed key, the sender combines its private key with the receiver's public key and the receiver combine its private key with sender's public key. For signature algorithm, the verification of the DSS signature differs from that of RSA. In RSA algorithm, the message digest is recovered from the signature and is then compared with the computed message digest. However, the sender's calculated message digest cannot be recovered when using DSS. Therefore, in DSS algorithm there is a computation based on the message digest as well as signature and then the receiver returns a yes or no answer for the verification.

Figure 3.3 shows the full handshake flow of DHE_DSS key exchange mechanism with server authentication. As we mentioned above, in order to generate pre-master secret, both server and client have to produce their DH keys and exchange the public DH keys to each other which can also be seen in Figure 3.3. And the server and the client generate the pre-master secrets with the peers' public keys as well as their own private keys. For authenticating the server, the server should generate a DSS signature. The client then verifies the signature with server's

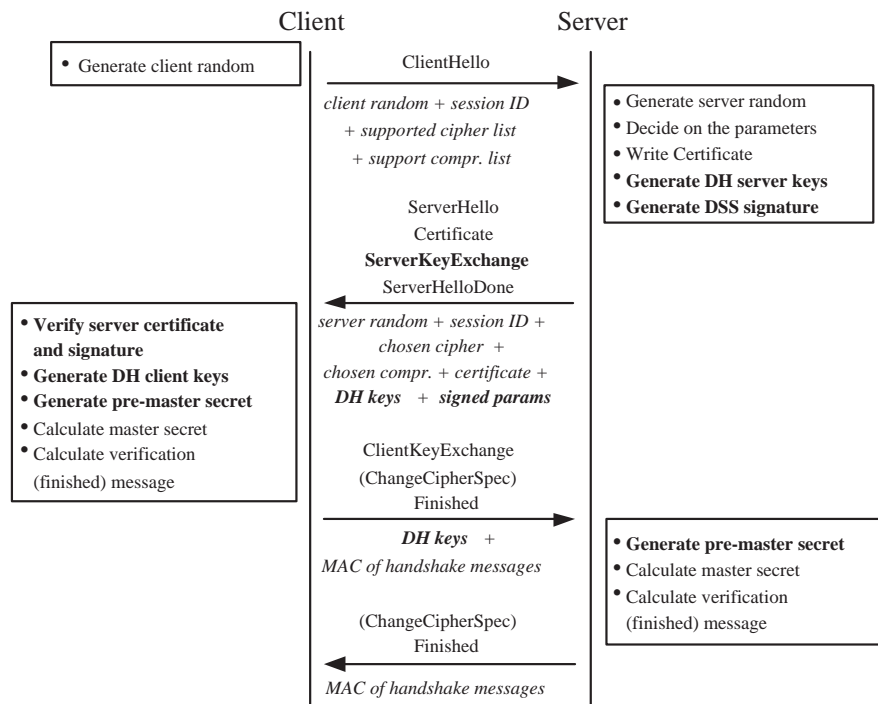


FIGURE 3.3: Full handshake for DHE_DSS using server authentication

DSS public key which is extracted from server's certificate. Figure 3.4 shows the DHE_DSS key exchange mechanism with mutual authentication. For achieving client authentication, the client needs to generate the certificate and the DSS signature. Then the server verifies the signature with client's public key which is extracted from client's certificate.

3.2 Pre-Shared Key Exchange based Mechanism

Pre-shared key exchange mechanisms for TLS protocol are proposed in [10] for the sake of supporting authentication based on pre-shared symmetric keys. Since these pre-shared keys are shared in advance to avoid public key operations, it is useful for performance-constrained environments and devices, e.g. mobile phones and Personal Digital Assistants (PDAs). In [10] 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.

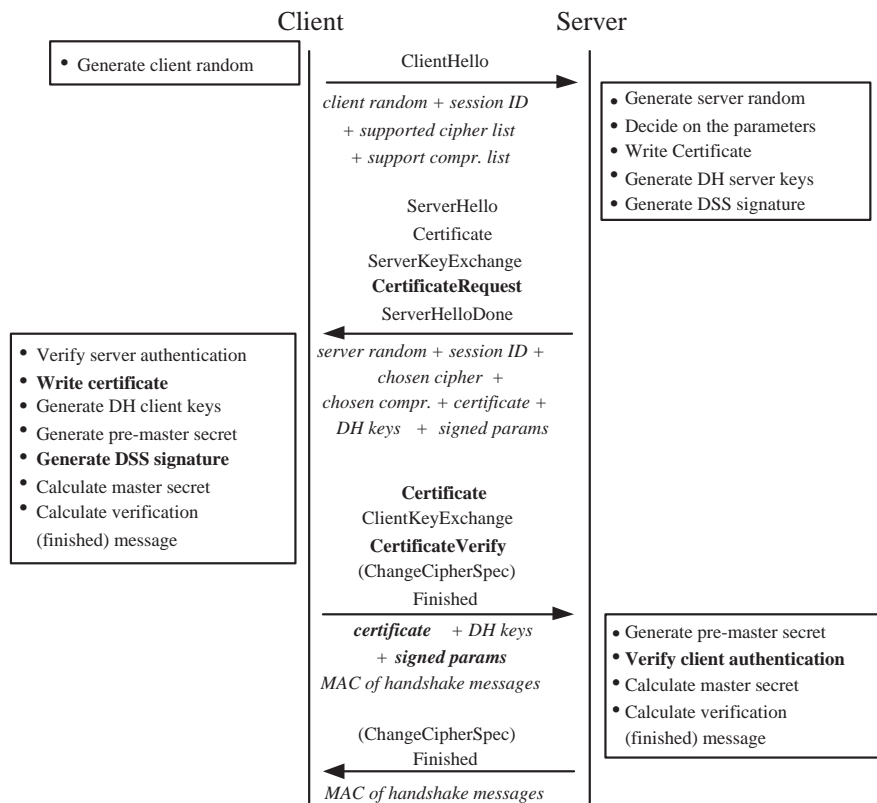


FIGURE 3.4: Full handshake for DHE_DSS using mutual authentication [3]

3.2.1 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 ??, 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.

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.

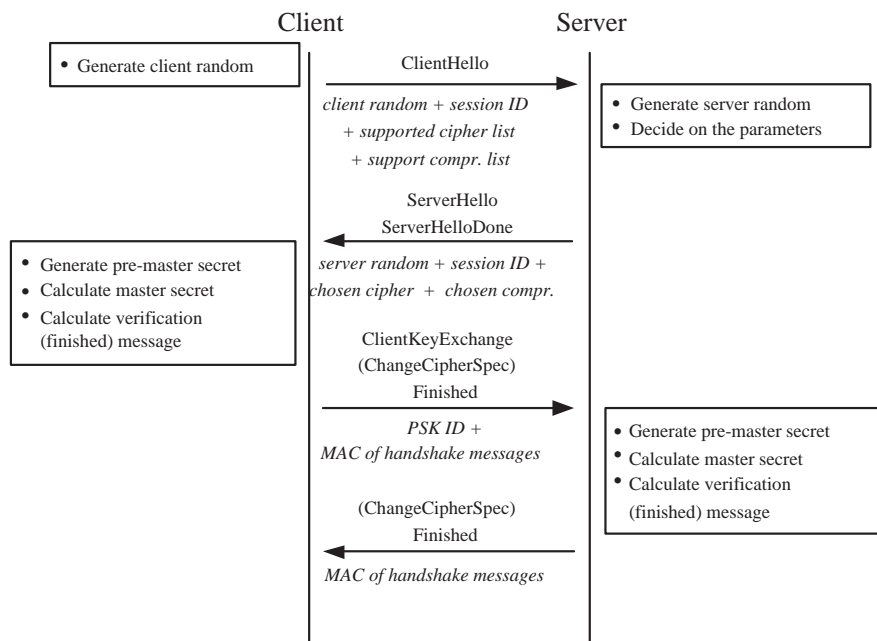


FIGURE 3.5: Full handshake for PSK

3.2.2 DHE_PSK

The mechanism of DHE_PSK is similar to that of DHE_DSS using server authentication, but utilizes the pre-share key (PSK) to authenticate a DH exchange instead of DSS signature. DH uses the **ServerKeyExchange** and **ClientKeyExchange** messages to transmit the DH parameters. As depicted in Figure ??, the server generates the DH server keys and writes 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, firstly, the normal DH computation is performed and then the pre-master secret is generated with the computed DH value and the pre-shared key.

3.2.3 RSA_PSK

The RSA_PSK key exchange mechanism uses RSA and certificate to authenticate the server and also achieves the mutual authentication by using PSK. As shown in Figure ??, 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

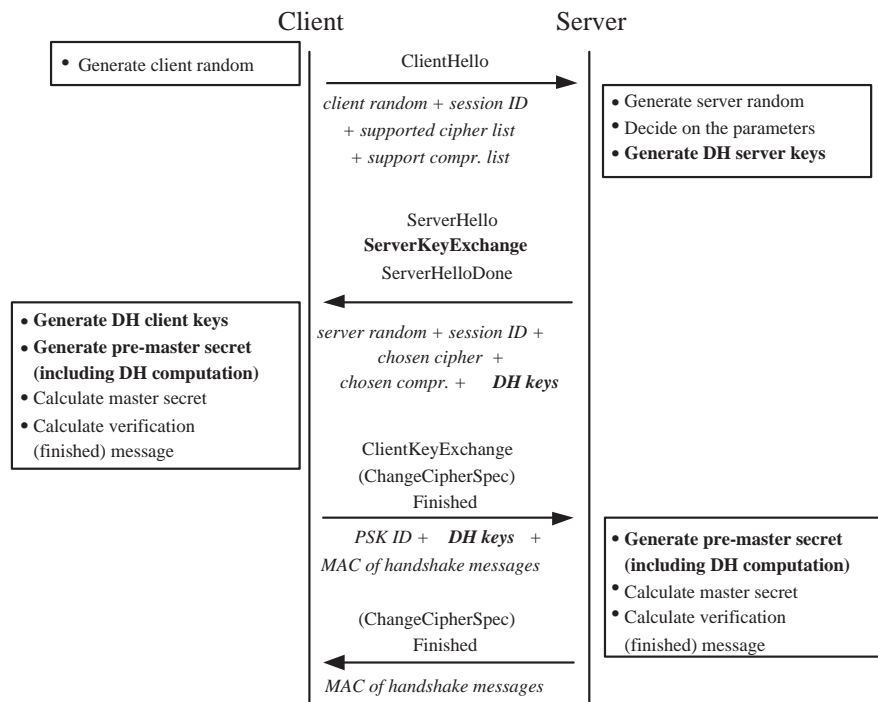


FIGURE 3.6: Full handshake for DHE_PSK

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.

3.3 Functional Comparison of Public and Pre-Shared Key Exchange Mechanisms in TLS

It has been proven that the public key exchange operations (encryption/decryption of the pre-master secret in RSA, generation of group keys in DHE, and the signing and verification for DSA or RSA signature) in TLS Handshake Protocol consume a significant amount of handshake duration [2]. As shown in Figure 3.5, the plain PSK uses only symmetric key algorithms, i.e. there is no burden caused by public key operations. Hence, the handshake time of plain PSK is even faster.

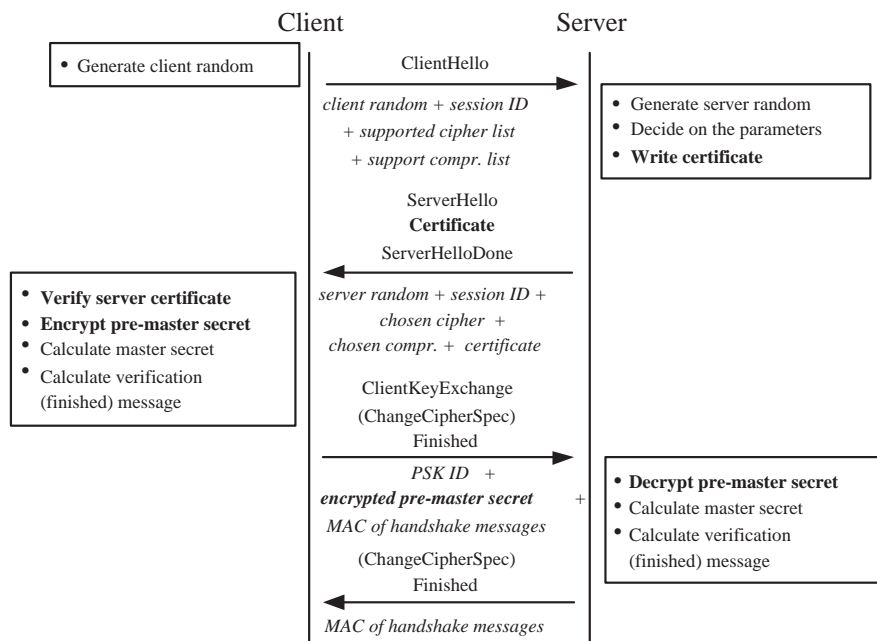


FIGURE 3.7: Full handshake of RSA_PSK mode

As we mentioned above, RSA_PSK is similar to the normal RSA using server authentication seen in Figure 3.7. However, RSA_PSK uses the pre-shared key and server certificate for mutual authentication while RSA using mutual authentication uses 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.

The mechanism of DHE_PSK is similar to that of DHE_DSS using server authentication. Nevertheless, DHE_PSK utilizes the pre-share key (PSK) to authenticate a DH exchange instead of DSS signature, i.e. no client and server certificates are transmitted and no digital digital signatures are operated during the TLS session when using DHE_PSK. Furthermore, among the ciphers based on PSK key exchange mechanism, only DHE_PSK can provide Perfect Forward Secrecy (PFS)¹.

¹In 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) which is 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 is static, the attacker might be able to break into the machine with the key and recover it.

Chapter 4

Performance Evaluation

4.1 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 of 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 handshake data
3. Handshake duration measurement

4.2 Experimental Setup

Our testbed consisted of two machines with fast processors as our fast server and fast client device and two devices with slow processors as our slow server and slow

client device. The fast server device is equipped with an Intel Pentium 4 CPU running at 2.53GHz with 1 GB of RAM and 80 GB HDD. The fast client device is equipped with an Intel Pentium 4 CPU running at 2.00GHz with 512 MB of RAM and 50 GB HDD. The slow client and the 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 - The Open Source toolkit for SSL/TLS.

4.3 Client and Server Performance

4.3.1 Plain PSK

Figure 4.1 and Figure 4.2 show the client and the server handshake processing time of the plain PSK, RSA using server authentication and DHE_DSS using server authentication in fast and slow processor devices, respectively. There are three types of processing time in device processing time, namely *key related operations*, *signature operations* and *common operations*. The item of key related operations means the processing time for key establishment, e.g. the generation of the PSK identity in client side when using plain PSK, the encryption of pre-master secret in client side when using RSA, and the computation of DH server group keys in server side when using DHE_DSS. Furthermore, the item of signature operations means the processing time for digital signature, e.g. verification of server RSA certificate in the client side using RSA mechanism. Finally, the item of common operations means the processing time for common operations in pre-shake and public key exchange mechanisms, e.g. the validation of the **Finished** message. Thus the common operation time in each mechanism is similar.

As seen in Figure 4.1 and Figure 4.2, 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 negligible small when compared to RSA and DHE_DSS., i.e. the plain PSK significantly reduces the client and the server processing time due to the lack of public key cryptographic operations. Furthermore, in the plain PSK mechanism, the difference of device processing time between the slow processor and the fast processor is smaller than that of the RSA mechanism or the DHE_DSS mechanism. In other words, the plain PSK is less sensitive to the processing

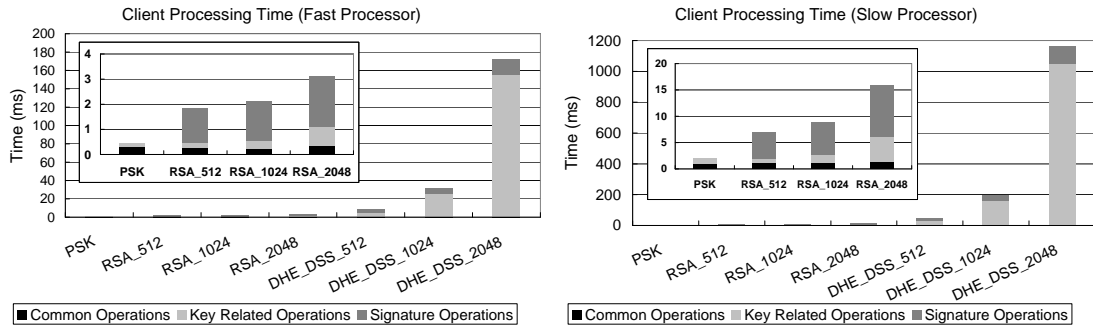


FIGURE 4.1: Client Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE_DSS using server authentication.

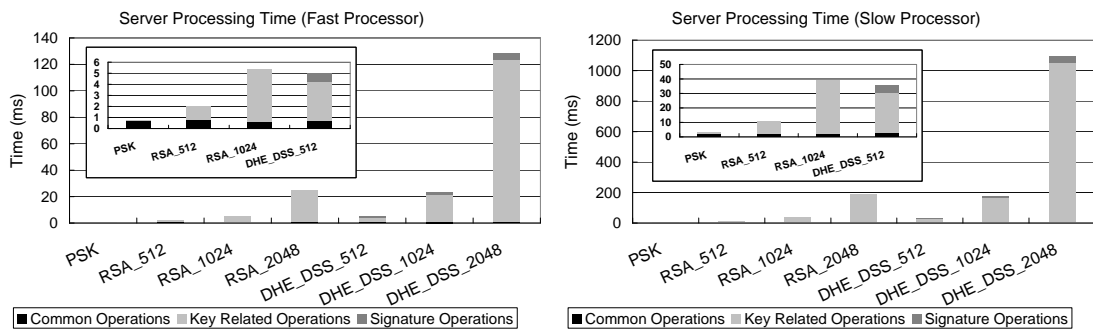


FIGURE 4.2: Server Processing Time in the handshake protocol when using plain PSK, RSA using server authentication and the DHE_DSS using server authentication.

capability of the devices. Accordingly, DHE_DSS is more sensitive to the device processing capability than RSA and the plain PSK because the device processing time of DHE_DSS increases largely when the same key size is used in powerless device.

4.3.2 DHE_PSK

Figure 4.3 and Figure 4.4 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 signature process while the mechanism of DHE_PSK is also similar to that of DHE_DSS but uses pre-shared key 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

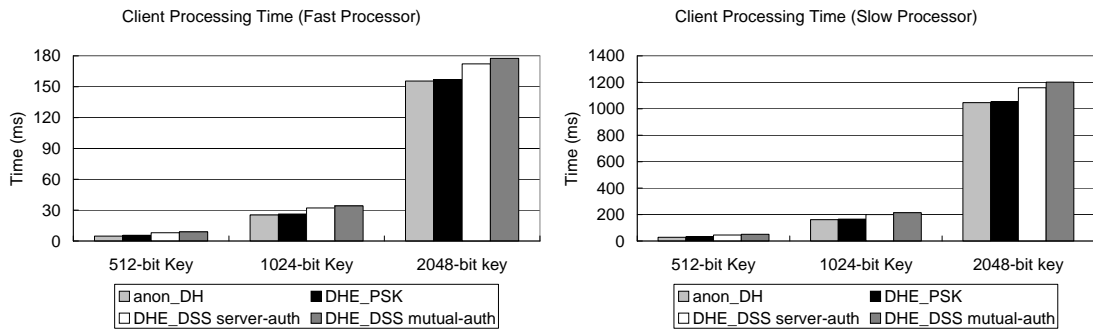


FIGURE 4.3: Client Processing Time in the handshake protocol when using anonymous DH, DHE_PSK, DHE_DSS using server authentication and the DHE_DSS using mutual authentication.

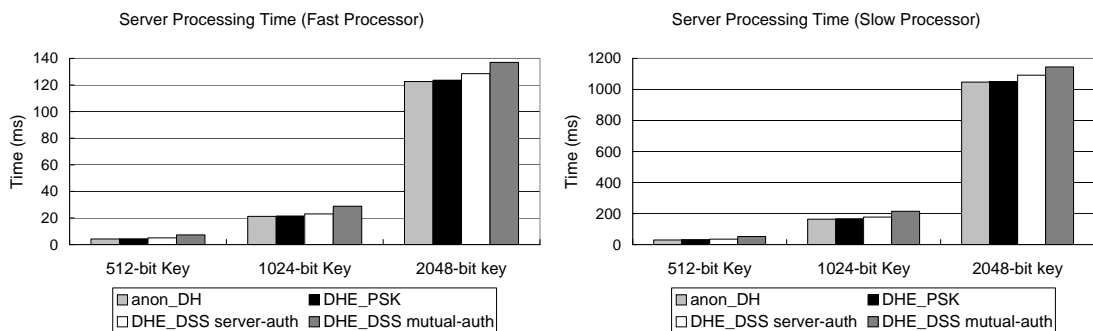


FIGURE 4.4: Server Processing Time in the handshake protocol when using anonymous DH, DHE_PSK, DHE_DSS using server authentication and the DHE_DSS using mutual authentication.

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 4.1 and Figure 4.2.

4.4 The Amount of transmitted Handshake Data

The transmitted handshake data amounts of different key exchange mechanisms are analyzed in Table 4.1. 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 using 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 mutual authentication need more data to be transmitted

TABLE 4.1: The amount of the 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

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.

4.5 Handshake Duration Analysis

After analyzing the device processing time and the amount of transmitted handshake data, we are going to analyze the impacts of those 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

$$T_{\text{Data Transmission}} = \frac{\text{Transmitted handshake data amount}}{\text{Network Throughput}} \quad (4.1)$$

4.5.1 Plain PSK

Figure 4.5 depicts the handshake duration of plain PSK, RSA and DHE_DSS. RSA and DHE_DSS use only server authentication. The evaluation scenarios are fast-server-to-fast-client and slow-server-to-slow-client, which represent best case and worst case, respectively. 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 transmitted data amount. Because the transmitted data amount of plain PSK is much less than that of RSA and

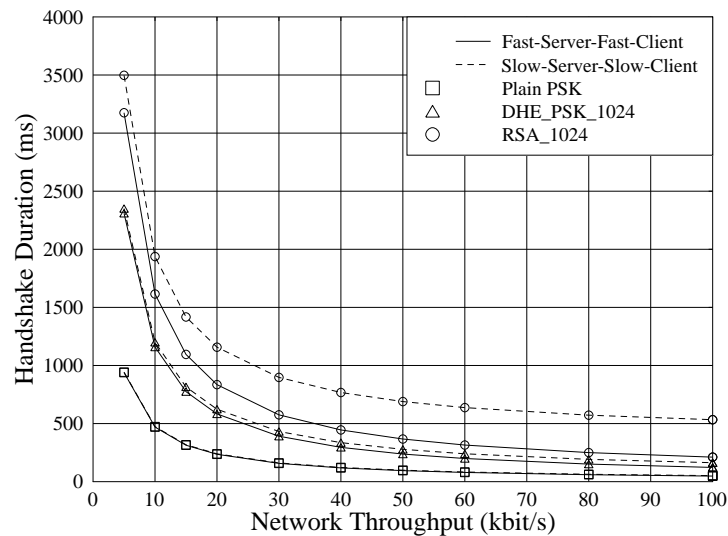
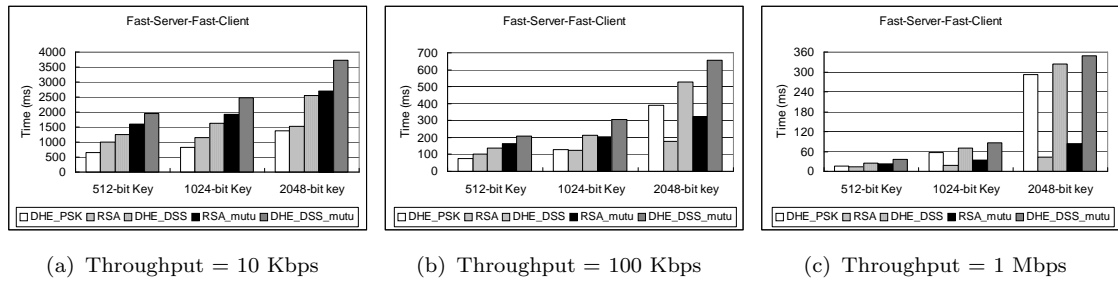


FIGURE 4.5: Handshake Duration of Plain PSK, RSA using server authentication, and DHE_DSS using server authentication in two scenarios, fast server to fast client and slow server to slow client. The key size for RSA and DHE_DSS is 1024-bit

DHE_DSS as illustrated in Table 4.1, the handshake duration of plain PSK increases more slowly when network throughput decreases as shown in Figure 4.5. Moreover, the difference of the handshake duration between the fast-server-fast-client and the low-server-to-slow-client scenario for plain PSK is slight while for DHE_DSS the difference is large. These properties are useful in constrained channel condition, e.g. wireless communication and even when using power constrained devices, e.g. handheld devices.

4.5.2 DHE_PSK

In Figures 4.6 - 4.7, 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 Figures 4.6 - 4.7, DHE_PSK always performs better than DHE_DSS using server or mutual authentication because DHE_PSK needs less device processing time and less amount of the transmitted data. Nevertheless, when compared to RSA, the handshake duration of DHE_PSK depends on device processing capability, key size and network condition. When using powerful processors with low network throughput, DHE_PSK performs better than RSA in terms of handshake duration as shown

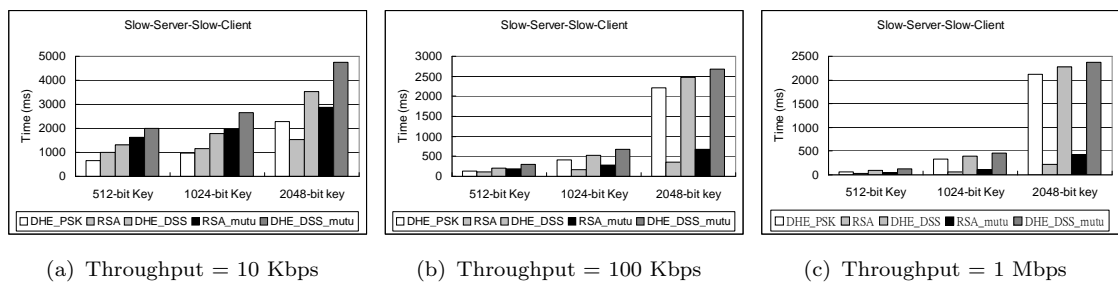


(a) Throughput = 10 Kbps

(b) Throughput = 100 Kbps

(c) Throughput = 1 Mbps

FIGURE 4.6: Handshake Duration of DHE_PSK, DHE_DSS using server authentication, DHE_DSS using mutual authentication, RSA using server authentication and RSA using mutual authentication in fast server to fast client scenario at 10k, 100k and 1M network throughput.



(a) Throughput = 10 Kbps

(b) Throughput = 100 Kbps

(c) Throughput = 1 Mbps

FIGURE 4.7: Handshake Duration of DHE_PSK, DHE_DSS using server authentication, DHE_DSS using mutual authentication, RSA using server authentication and RSA using mutual authentication in slow server to slow client scenario at 10k, 100k and 1M bps network throughput.

in Figure 4.6(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 4.6(b) - 4.6(c). When using powerless processors, DHE_PSK performs better than RSA mechanisms with small key sizes or when dealing with low network throughput environments as shown in Figure 4.7.

Note that the performance improvement achieved by RSA_PSK over RSA using mutual authentication should be similar to the improvement achieved by DHS_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 the comparison of DHS_PSK and DHE_DSS. However, the validity needs further in-depth research on RSA_PSK and RSA, which will be our future work.

Chapter 5

Conclusions

We have presented a systematic analysis of performance 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 less amount of transmitted data. However, when compared to RSA, DHE_PSK performs better than RSA only when using small key sizes or 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 securer communication among the pre-shared key based ciphers.

Bibliography

- [1] 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*, pp. 1245–1250, July 2003.
- [2] G. Apostolopoulos, V. G. J. Peris, and D. Saha, “Transport Layer Security: How Much Does It Really Cost?,” in *INFOCOM*, pp. 717–725, 1999.
- [3] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*. Boston, MA: Addison Wesley, 2000.
- [4] A. Frier, P. Karlton, and P. Kocher, “The Secure Socket Layer,” tech. rep., Netscape Communications Corp., November 1996.
- [5] T. Dierks and C. Allen, “The TLS Protocol Version 1.0,” rfc 2246, Internet Engineering Task Force, January 1999.
- [6] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, “SSH protocol architecture,” internet draft, Internet Engineering Task Force, Feb. 1999.
- [7] “Secure Electronic Transaction 1.0 specification,” May 1997. <http://www.setco.org>.
- [8] B. Ramsdell, “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification,” internet proposed standard rfc 3851, IETF, July 2004.
- [9] “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>.
- [10] P. Eronen and H. Tschofenig, “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS),” rfc 4279, Internet Engineering Task Force, Dec. 2005.

-
- [11] K. Kant, R. Lyer, and P. Mohapatra, "Architectural impact of secure socket layer on internet servers.," in *International Conference on Computer Design (ICCD)*, 2000.
- [12] C. Coarfa, P. Karlton, and D. Wallach, "Performance analysis of TLS Web Server," in *Proceedings of NDSS '02*, 2002.
- [13] 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.
- [14] P. G. Argyroudis, R. Verma, H. Tewari, and D. O'Mahony, "Performance Analysis of Cryptographic Protocols on Handheld Devices.," in *NCA*, pp. 169–174, 2004.
- [15] WAP Forum, "Wireless Transport Layer Security Specification," tech. rep., Wireless Application Protocol Forum Ltd., April 2001. <http://www1.wapforum.org/tech/documents/WAP-261WTLS-20010406-a.pdf>.
- [16] I. Herwono and I. Liebhardt, "Performance of WTLS and Its Impact on an M-commerce Transaction.," in *ICICS*, pp. 167–171, 2001.
- [17] R. L. Rivest, A. Shamir, and L. M. Adleman, "On Digital Signatures and Public Key Cryptosystems," technical report, MIT Laboratory for Computer Science, 1979.
- [18] National Institute of Standards and Technology (NIST), "Data Encryption Standard." U.S. Department of Commerce, December 1993.
- [19] R. Rivest, "The MD5 Message-Digest Algorithm," rfc 1312, IETF, April 1992.
- [20] H. Krawczyk, M. Ballare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," rfc 2104, IETF, February 1997.
- [21] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, 1976.
- [22] "Digital Signature Standard (DSS): FIPS-186." Federal Information Processing Standards Publication, May 1994.