On the Security of the TLS Protocol

Prastudy Fauzi

University of Tartu, Estonia
`prastudy.fauzi@gmail.com`

Abstract. In this research, the student will do a survey on TLS security, based on a recent paper by Krawczyk, Paterson, and Wee [KPW13]. Security is first shown for the most common mode of TLS, which is the TLS-RSA, under some security assumptions in the random oracle model. The security of other modes of TLS is then derived, and is shown to hold in the standard model.

Keywords: TLS, RSA

1 Introduction

TLS is one of the most important cryptographic protocols that we have today. It is used several times each day by internet users to achieve secure communication over the network. The security of TLS is thus a very significant issue both in theoretical and applied cryptography. However, due to the nature of TLS, it has been very difficult to prove its security. In fact, it is insecure if we use common security definitions of key exchange protocols. The main problem is the interleaving property of the two parts of TLS: the TLS Handshake Protocol, and the TLS Record Protocol. Moreover, the main mode of TLS is TLS-RSA using PKCS #1 v1.5, which is not CCA-secure: Bleichenbachers attack worked on SSL, but only just thwarted in TLS. The security of TLS will thus need to be addressed with these things in mind.

We will do a survey of a recent paper by Krawczyk, Paterson, and Wee, which provides a detailed analysis of TLS security using reasonable security assumptions. We start this survey by giving some basic definitions in cryptography. We then summarize the steps used in the paper to achieve their goal of proving TLS security. First, the TLS Handshake protocol is simplified by removing some messages and abstracting some implementation details. This will then enable a reasonable security definition for the TLS Protocol, i.e. Authenticated Confidential Channel Establishment (ACCE) security. Second, the most common mode of the TLS Handshake Protocol is shown to be insecure under the usual security definitions, and this will motivate the introduction of Constrained Chosen Ciphertext Attack (CCCA) security. Third, this modified security definition is shown to be sound, as the CCCA security of the TLS Handshake Protocol used with a stateful Length-Hiding Authenticated Encryption (sLHAE) scheme in the TLS Record Protocol will result in an ACCE-secure TLS Protocol. Finally, the CCCA security of TLS Handshake Protocol is proved to hold in the TLS-RSA mode, and the proof extends to other modes. This will complete the proof, as the sLHAE security of the TLS Record Protocol has already been known to hold.

2 Preliminaries

We will give an overview of the basic cryptographic concepts used in the later sections.

2.1 Public Key Encryption

A public key encryption (PKE) scheme consists of three elements: key generation, encryption, and decryption.

– Key generation is the process of generating a public key and secret key pair for encryption and decryption. It requires a security parameter λ, typically the size of the resulting public key.
– Encryption is the function that maps a plaintext into a ciphertext, using a public key. The domain of this encryption function is called plaintext space.
– Decryption is the function that maps a ciphertext back into plaintext, using a secret key.

A PKE scheme with a key generation algorithm KG, encryption algorithm E and decryption algorithm D can then be written as (KG, E, D). Some well-known PKE schemes include RSA, ElGamal and Paillier.

2.2 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) $(KeyGen, Enc, Dec)$ consists of three elements: key generation, encapsulation, and decryption.

– Key generation is the process of generating a public key and secret key pair $(PK, SK) \leftarrow KeyGen(1^\lambda)$ for encryption and decryption, similar to that in PKE.
– The encapsulation algorithm creates a symmetric key K together with a ciphertext c.
– Decryption is the function that, given a ciphertext c, outputs a key $K \leftarrow Dec(SK, c)$.

Additionally, a labeled KEM (LKEM) is a KEM with the addition that Enc and Dec take as additional input a label $L \in \{0,1\}^*$, such that if $(K, c) = Enc(PK, L)$ then $Dec(SK, L^*, c) = K$ if and only if $L = L^*$.

2.3 IND-CCA2 Security

For a PKE scheme (KG, E, D), CCA2 is an adaptive chosen-ciphertext attack in which an attacker, given a ciphertext $c = E(m_b)$ can send a number of ciphertexts $c_1, c2, \cdots c_j$ to be decrypted, and may use these decryption results to query more ciphertexts $c_{j+1}, \cdots$. The attacker uses these results to determine if c was an encryption of m_0 or an encryption of m_1. The decryption is done as long as $\forall i, c_i \neq c$.

A PKE scheme is indistinguishable under adaptive chosen-ciphertext attack (IND-CCA2-secure, or sometimes just known as CCA-secure) if for all polynomial-time adversaries $\mathcal{A}$, the probability that he wins the security game G, $Pr[G^{\mathcal{A}} = 1]$ is negligible. In this case, G is defined as follows:

```
$G$
    $PK, SK \leftarrow KG(1^\lambda)$
    $m_0, m_1 \leftarrow \mathcal{A}^{D(\cdot)}(PK)$
    $b \xleftarrow{\$} \{0,1\}$
    $c \leftarrow E(m_b)$
    $b' \leftarrow \mathcal{A}^{D(\cdot)}(PK, c)$
    $return\ b = b'$ .
```

Here $\mathcal{A}^{D(\cdot)}$ means that the adversary gets access to a decryption oracle, which outputs the correct decryption whenever the queried value is not c. In essence, CCA security implies that for any two messages m_0, m_1, an adversary cannot distinguish between them, except with a very small probability.

2.4 Stateful Encryption

A stateful authenticated encryption with associated data (AEAD) scheme
$stE = (stE.Gen, stE.Init, stE.Enc, stE.Dec)$ consists of four algorithms as follows:

– $stE.Gen$ generates a symmetric key K from the key space.
– $stE.Init$ initializes two states ST_e, ST_d, where ST_e is used for encryption and ST_d is used for decryption.
– $stE.Enc$ gets as input a key K, desired output length ℓ, associated data H, message m and encryption state ST_e and outputs a ciphertext c and the new encryption state ST'_e
– $stE.Dec$ is a decryption function defined similarly as $stE.Enc$, but does not need ℓ.

A stateful AEAD scheme is stateful Length-Hiding Authenticated Encryption (sLHAE) if it is secure against the adaptive chosen-ciphertext attack described previously, but in the stateful encryption setting. This would also imply non-malleability of stE.

The most common sLHAE-secure stateful AEAD encryption schemes are (symmetric) block ciphers which use MAC-then-Encrypt.

3 Security of TLS

In this section we will analyse various aspects of the TLS protocol, with the ultimate goal of proving its security under reasonable assumptions. We will focus on the most common mode of TLS used today, which is TLS-RSA with server-only authentication. We start with defining the security goal, which is (S)ACCE security, and define security notions related to the TLS Record and Handshake protocols. We will then argue that TLS is secure under these notions, and finally show that these notions can be proved to hold with additional security assumptions.

3.1 TLS Protocol

Transport Layer Security (TLS) [DR08], is a successor to Secure Sockets Layer (SSL), which was designed by the web browser company Netscape in 1994 to provide secure communication over the Internet. It is used not only to browse web pages securely using HTTPS, but also for other common applications such as VPN, instant messaging, and Dropbox. As such, it is the most widely used cryptographic protocol over computer networks.

TLS consists of two protocols, TLS Handshake Protocol and TLS Record protocol. Some of the difficulties in proving the security of TLS arises from the fact that these two protocols are interleaving: some of the messages in the Handshake protocol are encryptions using the to-be-agreed-upon session / application key, as can be seen in Figure 1.

Krawczyk, Paterson, and Wee [KPW13] defined four main TLS modes:

– TLS-RSA: uses the RSA PKCS #1 v1.5 PKE scheme, and is the most widely used mode.

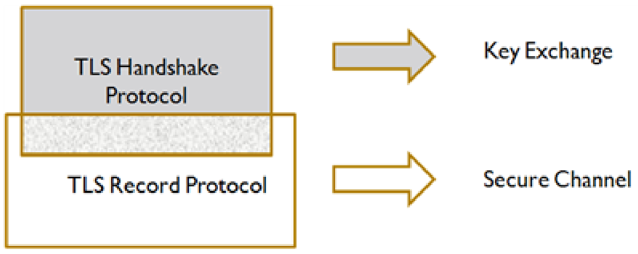

Fig. 1. Interleaving property of TLS.

- TLS-CCA: this is an ideal setting where a CCA-secure PKE scheme is used, for example RSA-OAEP.
- TLS-DH and TLS-DHE: these modes use Diffie-Hellman keys either getting a Diffie-Hellman key exchange, or using an ElGamal encryption scheme.

TLS Handshake Protocol The TLS Handshake Protocol is done by a client and authenticated server with the goal of achieving a common application key $AKEY = CKEY||SKEY$. Krawczyk, Paterson, and Wee [KPW13] did a simplification of the Handshake protocol such that some messages such as the $ChangeCipherSpec$ and session resumption are removed to achieve a more flexible proof. This simplification can be seen in Figure 2.

Formally, the client and server do a 4 round protocol as in Figure 3.

The functions can be defined as follows. We have abstracted away many parts, but one thing to note is that the algorithms $tls.Enc, tls.Dec$ make use of a secure pseudo-random function with the shared string η in the key, plus a random key derivation function to achieve private values $AKEY, USFIN, UCFIN$ computable by both honest client and server. Hence we assume the existence of a secure pseudo-random function family.

```
$Client$
    $\eta_C \leftarrow \{0,1\}^\lambda$
    $CREQ \leftarrow \eta_C$
    **return** $CREQ$

$Server(CREQ)$
    $\eta_S \leftarrow \{0,1\}^\lambda$
    $CERT_S \leftarrow IS_S, PK_{SCA}$
    **return** $SRES \leftarrow \eta_S||CERT_S$

$Client(SRES)$
    if $Check(CERT_S) fails$ then $abort$
    $\eta \leftarrow \eta_C||\eta_S$
    $(CRES, AKEY||UCFIN||USFIN) \leftarrow tls.Enc(PK_s, \eta)$
    $(ST_e^C, ST_d^C) \leftarrow stE.Init(1^\lambda)$
    $CFIN \leftarrow stE.Enc(CKEY, \ell_C, H_C, UCFIN, ST_e^C)$
    **return** $CRES, H_C, CFIN$
```

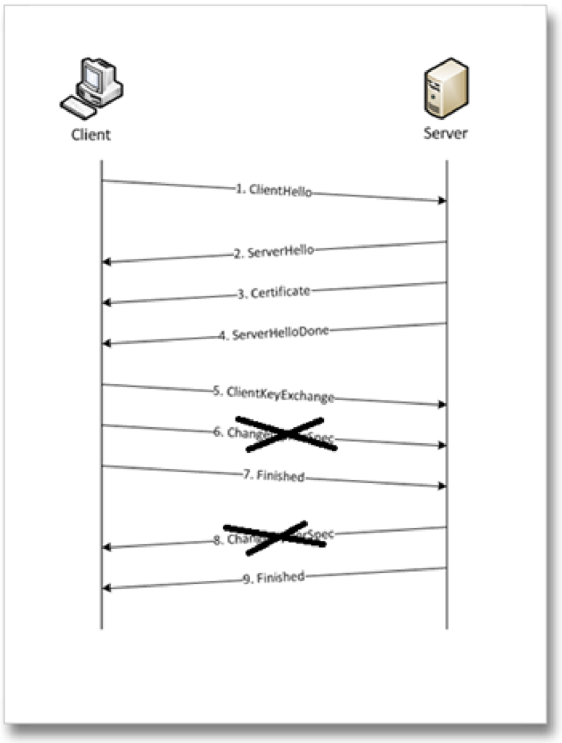

Fig. 2. Simplified version of the TLS Handshake Protocol with server-only authentication. Taken from https://devcentral.f5.com/articles/ssl-profiles-part-1

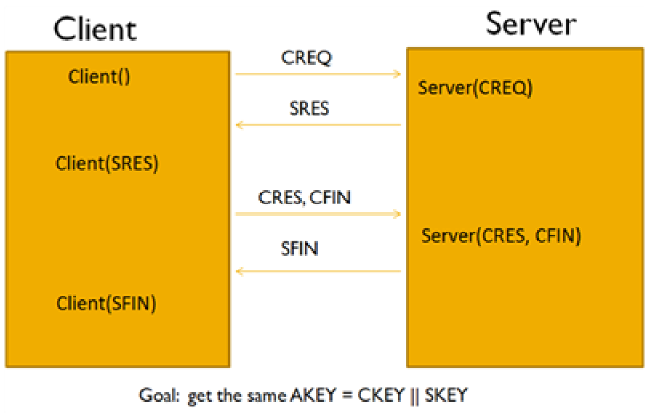

Fig. 3. Abstraction of TLS Handshake.

```
$Server(CRES, H_C, CFIN)$
    $\eta \leftarrow \eta_C || \eta_S$
    $AKEY || UCFIN || USFIN \leftarrow tls.Dec(SKS, \eta, CRES)$
    if $tls.Pred(AKEY || USFIN || UCFIN, CFIN) = 0 \;\; \Lambda = rej; \;\; abort$
    $SFIN \leftarrow stE.Enc(SKEY, \ell_S, H_S, USFIN, ST_e^S)$
    $\Lambda = acc$
    **return** $H_S, SFIN$
```

```
$Client(H_S, SFIN)$
    $Check\ \ stE.Dec(SKEY, H_S, SFIN, ST_d^C) = USFIN$
    if $fail, \Lambda = rej;\ \ abort$
    else $\Lambda = acc$
```

ACCE Security An Authenticated Confidential Channel Establishment Π is a client-server protocol which is an even higher-level abstraction of the TLS Handshake Protocol, such that when a party reaches the state $\Lambda = acc$, it has executed the initialization of the stateful encryption scheme $stE.Init$ somewhere in the protocol run, and it has already generated the key $AKEY = CKEY||SKEY$ which will be the shared session key between the client and server. The client must use $SKEY$ in $stE.Enc$ to send data to the server, while the server must use $CKEY$ in $stE.Enc$ to send data to the client.

From this abstraction, we can define Server-only Authenticated Confidential Channel Establishment (SACCE) security, which has two parts

– Server authentication: if the client reaches state $\Lambda = acc$, then the client and server must have matching keys $AKEY$. The probability that an adversary can compromise server authentication is $\mathsf{Adv}_{\Pi}^{sa}(\mathcal{A})$.
– Channel security: no other party can compromise $AKEY$. The probability that an adversary can compromise channel security is $\mathsf{Adv}_{\Pi}^{cs}(\mathcal{A})$.

The motivation of having it server-only is because TLS is mostly used with server-only authentication. More formally, an ACCE protocol Π is SACCE-secure if both Adv_{Π}^{sa} and Adv_{Π}^{cs} are negligible.

The main question can then be whether or not TLS-RSA with RSA PKCS #1 v1.5 encryption is SACCE-secure.

3.2 CCCA Security

We now look at the what security notion the TLS Handshake Protocol must achieve to get our goal of SACCE security for TLS.

Bleichenbacher's Attack Bleichenbacher gave an attack [Ble98] on SSL, such that when an adversary is given a ciphertext validity oracle, then after several queries to this oracle, the session key is compromised. This made use of the weakness of PKCS #1 v1.5, which is not CCA-secure, and the fact that there are two types of error messages that a server can give.

TLS Fix from SSL A logical way to fix this would be to force the use of a CCA-secure PKE scheme, i.e. the TLS-CCA mode, or the other Diffie-Hellman based schemes TLS-DH and TLS-DHE. However, practitioners instead fixed the server's error messages so that there is only one type of error message. In this way, an adversary cannot learn if an abort happened because of an invalid ciphertext, or because of a valid ciphertext with invalid Finished ($CFIN$ or $SFIN$) message.

Constrained CCA The fix above can be formulated as Constrained CCA (CCCA) security. In this setting, the adversary in a CCA game is not given a decryption oracle, but rather a constrained decryption oracle, as follows. The oracle takes as input a ciphertext C and tag T, and will only return the corresponding plaintext if $pred(M, T) = 1$.

```
$CDEC(C, T)$
    $M \leftarrow Dec(sk, C)$
    if $pred(M, T) = 0$ return $\perp$
    else return $M$
```

It is clear that this corresponds to the TLS fix for Bleichenbacher's attack, wehre the predicate $pred$ corresponds to checking the validity of the message M.

We can now define CCCA security as follows: a KEM is CCCA-secure if for all polynomial-time adversaries $\mathcal{A}$, there is negligible probability of winning the CCCA game defined as follows:

```
$G$
    $PK, SK \leftarrow KG(1^\lambda)$
    $m_0, m_1 \leftarrow \mathcal{A}^{CDEC(\cdot)}(PK)$
    $b \leftarrow \{0, 1\}$
    $c \leftarrow E(m_b)$
    $b' \leftarrow \mathcal{A}^{CDEC(\cdot)}(PK, c)$
    return $b = b'$ .
```

3.3 Proving CCCA + sLHAE implies SACCE

Theorem 1. *[KPW13] If the LKEM tlskem used in TLS is IND-CCCA-secure and stE is sLHAE-secure, then TLS is SACCE-secure.*

The proof idea is that IND-CCCA security implies that an adversary cannot distinguish between an encryption of $AKEY||USFIN||UCFIN$ and a random encryption.

First we use this to show that an adversary cannot distinguish between an encryption of $AKEY||USFIN||UCFIN$ and an encryption of $AKEY||USFIN'||UCFIN$. Therefore an adversary cannot come up with a corresponding $SFIN$ value except with negligible probability. Hence, $\mathsf{Adv}^{sa}_{TLS}(\mathcal{A})$ is negligible.

Second we use this to show that an adversary cannot distinguish between an encryption of $AKEY||USFIN||UCFIN$ and an encryption of $AKEY'||USFIN||UCFIN$. Hence we have a random $AKEY$ that is used in stE which is sLHAE-secure. Moreover stE is also non-malleable so an adversary cannot make a fake $CFIN$. So, $\mathsf{Adv}^{cs}_{TLS}(\mathcal{A})$ is also negligible. By definition, TLS is SACCE-secure.

3.4 Proving CCCA Security

Next, we cite a known result about the security of the TLS Record Protocol by Krawzyk.

Theorem 2. *[Kra01] The stateful encryption scheme stE is sLHAE-secure.*

Finally, SACCE security is completed by the following theorem [KPW13]

Theorem 3. *If the KEM is OW-PCA, the PRF used in $tls.Enc, tls.Dec$ is a pseudo-random function, and the key derivation function Kdf can be modeled as a random oracle, then the LKEM tlskem with predicate $tls.Pred$ is IND-CCCA-secure.*

An additional assumption, one-way secure under plaintext-checking attacks (OW-PCA), is needed here, and it is equivalent to a stronger version of the RSA assumption. The proof idea is to simulate the $CDEC$ oracle using a random oracle, without knowing the secret key. Due to complexity, we omit the proof.

One important note here is that the random oracle and OW-PCA assumptions are only required because we use the weak TLS-RSA mode. These are not needed in the security proof of the other RSA modes. Hence this shows that TLS-RSA is in fact SACCE-secure in the random oracle model, while the other modes are SACCE-secure in the standard model.

4 Discussion

The omission of negotiation messages from the TLS model used for analysis means that ciphersuite downgrade attacks and protocol version rollback attacks are not covered. It is interesting to see if TLS is still secure after adding these back into the protocol, as not all the additions are trivial. It must be noted that the other well-known attacks of TLS such as the BEAST (Browser Exploit Against SSL/TLS) attack are only possible because of a bad choice and implementation of TLS symmetric encryption algorithms. Correct implementations of the CBC-mode is known to be $sLHAE$-secure. One other problem in this case is that the OW-PCA assumption is an even stronger assumption than the RSA assumption.

We note that although TLS-RSA is the most common mode of TLS used, it is the least secure, and not even the most efficient. One obvious way to improve the security of TLS is to force the use of CCA-secure PKE schemes, as even a slight change from RSA to RSA-OAEP will increase the confidence in security significantly. Another good idea would be to fix the TLS protocol itself such that the Handshake and Record Protocols are separated. This will significantly simplify the security analysis of TLS, and ultimately increase the confidence in the protocol.

Finally, though we focus here in server-only ACCE, the results hold if the client is also authenticated. Hence TLS is also ACCE-secure with basically the same assumptions.

5 Summary

Proving the security of TLS is challenging because of the overlap between the TLS Handshake and Record Protocols. We have shown that ACCE is a good security definition that accurately describes the properties of TLS. Moreover, we showed that TLS-RSA is CCCA-secure in the random oracle model, while the other modes TLS-CCA, TLS-DH, TLS-DHE are CCCA-secure in the standard model. This was used, in conjunction with a previously known result about the security of the TLS Record Protocol if we use sLHAE-secure encryption scheme, to prove that TLS is ACCE-secure in all the stated TLS modes.

There is still room for improvement, as the analysis does not include adversaries who use session resumption or ciphertext renegotiation attacks.

References

Ble98. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO '98*, volume 1462, pages 1–12, Santa Barbara, California, USA, August, 23–27 1998.

DR08. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. Technical Report 5246, RTFM, Inc., August 2008. Available at http://tools.ietf.org/html/rfc5246.

KPW13. Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the Security of the TLS Protocol: A Systematic Analysis. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, volume 8042, pages 429–448, Santa Barbara, California, USA, August, 18–22 2013.

Kra01. Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In Joe Kilian, editor, *CRYPTO 2001*, volume 2139, pages 310–331, Santa Barbara, California, USA, August, 19–23 2001.