Universally Composable Security Analysis of TLS*

Sebastian Gajek[1], Mark Manulis[2], Olivier Pereira[2], Ahmad-Reza Sadeghi[1], and Jörg Schwenk[1]

[1] Ruhr University Bochum, Germany
sebastian.gajek|joerg.schwenk@nds.rub.de
ahmad.sadeghi@trust.rub.de
[2] Université Catholique de Louvain, Belgium
mark.manulis|olivier.pereira@uclouvain.be

Abstract. We present a security analysis of the complete TLS protocol in the Universal Composable security framework. This analysis evaluates the composition of key exchange functionalities realized by the TLS handshake with the message transmission of the TLS record layer to emulate secure communication sessions and is based on the adaption of the secure channel model from Canetti and Krawczyk to the setting where peer identities are not necessarily known prior the protocol invocation and may remain undisclosed. Our analysis shows that TLS, including the Diffie-Hellman and key transport suites in the uni-directional and bi-directional models of authentication, securely emulates secure communication sessions.

Keywords: Universal Composability, TLS/SSL, key exchange, secure sessions

1 Introduction

The protocol framework of *Transport Layer Security (TLS)* [1] serves as fundamental primitive for WWW security and has fostered to the most valuable cryptographic protocol family in practice. The TLS protocol suites enable applications to communicate across a distributed network in a way that endpoint authentication and transmission privacy is guaranteed. The main goal of this paper is to provide a rigorous and generic analysis of TLS's cryptographically relevant parts of the protocol framework, namely the handshake and record-layer protocols. Given the wide deployment of TLS and the fact that it has been designed as contemporary cryptography started to explore provable security, it is natural that this analysis is of high, practical interest. Since TLS has already been investigated with respect to certain cryptographic primitives and protocol abstractions (see below), a general belief is that the framework is secure. Yet, there is no security proof of the entire TLS protocol and a careful observation

* A full version of this paper is available at http://eprint.iacr.org/2008/251.

of TLS's subtleties in the various modes provided by the different cipher suites. However, such a proof would significantly contribute to the analysis of complex protocols executed on top of TLS.

Our analysis is carried out in the meanwhile classical model of Universally Composable (UC) security [2] which guarantees protocol security under general composition with arbitrary other protocols. This valuable property stimulated the search for universal protocol design techniques and their realizations [3–8]. On the other hand, there are important impossibility results [3, 9] so that a security proof of TLS in this model is neither obvious nor trivial. Our work particularly continues the way of Canetti's and Krawczyk's consideration of the Σ-protocol underlying the signature based modes in IPSec [10] and their model to build up secure channels [11] in the UC model with the exception that instead of proving single modes, we utilize UC as technique to prove the complete protocol secure in a single proof. Applied to the analysis of TLS, it includes Diffie-Hellman and encrypted key transport in the uni- or bi-directional model of authentication which are part of the TLS handshake, and their emulation to build secure communication protocols realized by the additional TLS record layer.

The most relevant question is how to reduce the complexity of the proof. Is it possible to unitize TLS in meaningful protocol fragments such that the composition theorem allows for an efficient protocol reformulation in the hybrid model? That means, can we define ideal functionalities that capture the cryptographic task of some of its fragments and simply reuse these functionalities with the next fragment? Otherwise, a composite analysis would not make sense so that we could switch to stand-alone protocol proofs. Fortunately, we answer the questions in the positive. To this end, we introduce two ideal functionalities, dubbed the *universal key exchange* and *universal secure communication sessions*. The functionalities are "universal" in the sense that they emulate different key establishment methods and modes of authentication in a self-contained definition. We show that the TLS framework including the different modes securely emulates the universal secure sessions functionality in the presence of *non-adaptive adversaries*. Our result can significantly simplify security proofs of higher-layer protocols by employing the composition theorem. We are not aware of any prior work that evaluates the essential composability property of TLS.

Related Work Because of its eminent role the TLS framework has been repeatedly peer-reviewed. Schneier and Wagner [12] gave the first informal analysis in the core specification. Bleichenbacher [13] found some weaknesses in the PKCS#1 standard for RSA encryption as used with some SSL 3.0 handshake protocols.[3] Jonsson and Kaliski [14] showed that the encryption in the revised PKCS#1.5 standard is secure against chosen cipher attacks in the Random Oracle Model. Krawczyk [15] analyzed the composition of symmetric authentication and encryption to establish a secure communication channel with TLS record

[3] Note that the attack exploited weaknesses of the PKCS#1 standard and not the TLS protocol.

layer protocols and found some problems in the case of general composition. However, these do not apply to the standard cipher suites.

Apart from the analysis of some cryptographic primitives, a line of research addressed the analysis of *dedicated* TLS protocols on the basis of cryptographic abstractions to allow automated proof techniques. Paulson [16] gave an inductive analysis of a simplified version of TLS, using the theorem proving tool Isabelle. Mitchell, Shmatikov, and Stern [17] checked TLS, using the finite-state enumeration tool named Murphϕ. Ogata and Futatsugi [18] used the interactive theorem prover OTS/CafeObj to check a simplified version of the key transport handshake protocol through equational reasoning. He *et al.* [19] provided a proof of correctness of TLS in conjunction with the IEEE 802.11i wireless networking protocol, using the Protocol Composition Logic. The drawback these tool-supported approaches currently share is that the proofs are considerably simplified. They follow the *Dolev-Yao model* [20] which represents cryptography as term algebras and abstracts away the comprehensiveness of the adversary such that the proofs are not known to be cryptographically sound.

Very recently, Morrissey *et al.* [21] analyzed in an independent work the modularity of a *TLS-related* handshake protocol in a game-based style. The handshake is not exactly conform with the core TLS specification [1] and considers not all protocol variants. Their work focuses on a generic proof of the iterated session key constructions. By contrast, our work is of independent interest and practical relevance. We investigate TLS's intrinsic compositional property which is to provide higher-layer protocols with some secure communication functionality. Furthermore, our work addresses the native handshake protocols and additionally the record layer protocols in different authentication models under the stronger security notion of universally composable security.

Organization The remaining sections are structured as follows. Section 2 clarifies notation and cryptographic building blocks. Section 3 shortly introduces the TLS protocol family and describes the compositional proof idea. Section 4 is devoted to the TLS handshake subroutines we use throughout the analysis. Section 5 proves the full framework and Section 6 concludes.

2 Preliminaries

2.1 Notations

The protocols run between two players: a client and a server. A player P may act as *initiator* I or *responder* R. If P is acting as I then by $\bar{P}$ we denote a player acting as R and viceversa. An anonymous player, i.e. a party whose identity is not known is denoted by $\perp$. We refer to the handshake protocol structure as π and the composition with the record-layer protocols as ρ. Additionally, we use different indices to capture the modes of authentication in ideal functionalities. We refer to a responder-only authenticated functionality as $\mathcal{F}^1$, i.e. a functionality where the responder authenticates to the initiator, but the initiator's identity

remains unknown. Further, we denote an ideal functionality, where both players authenticate by $\mathcal{F}^2$, and a hybrid functionality of $\mathcal{F}^1$ and $\mathcal{F}^2$ by $\mathcal{F}^{(1,2)}$.

2.2 Cryptographic Building Blocks and their Constructions

The specification of TLS [1] uses several cryptographic primitives and mandates or recommends certain instantiations of them as described in the following:

An ASYMMETRIC ENCRYPTION SCHEME $(\mathtt{ENC}_{pk_R}(), \mathtt{DEC}_{sk_R}())$ for transporting the encrypted premaster secret which must be instantiated with the RSA-OAEP construction (known to provide indistinguishability under adaptive chosen ciphertext attacks [14] in the Random Oracle Model). In TLS handshake a private key sk_R is known to the responder R and its public key pk_R is signed by a Certification Authority (CA).

A DIGITAL SIGNATURE SCHEME $(\mathtt{SIG}_{sk}(), \mathtt{VER}_{vk}())$ for entity authentication which can be instantiated with DSA and RSA-PSS (the latter is known to provide weak existential unforgeability under chosen message attacks in the Random Oracle Model [22]). Each player owns a signing key sk and the respective verification vk is certified by a CA.

A MESSAGE AUTHENTICATION CODE function $\mathtt{HMAC}_k()$ from [23] and a SYMMETRIC ENCRYPTION SCHEME $(\mathtt{E}_k(), \mathtt{D}_k())$ which is recommended to be DES or 3DES in different modes and with different key lengths. The construction of symmetric authentication *with* encryption is known to provide weak unforgeability under chosen message attacks and indistinguishability under chosen plaintext attacks [15, 24].

A PSEUDO-RANDOM FUNCTION for the key derivation and confirmation, denote here by $\mathtt{PRF}_k()$. It is evaluated with seed k on an input string l_i, $i \in [1, 4]$ which is labeled with different publicly known space delimiters and two independently chosen random values, i.e. the nonces exchanged in the first protocol, or a function thereof. The specification defines a special construction based on HMAC combiners which has been recently proven to be a good randomness extractor [25].

3 Transport Layer Security

3.1 TLS in a Nutshell

The standard TLS specification [1] comprises handshake, alert, change cipher spec, and record layer (sub)protocols. The *handshake protocol* is used to negotiate key material and cryptographic algorithms and the record layer protocol can then be applied to secure transmitted application data. The *change cipher spec protocol* consisting of one message triggers a change in the cryptographic parameters used by the record layer, while the *alert protocol* communicates error messages, whenever a failure during the handshake or message protection occurs. Thus, the essential cryptographic building blocks for TLS and target to the presented analysis are the handshake and record layer protocols.

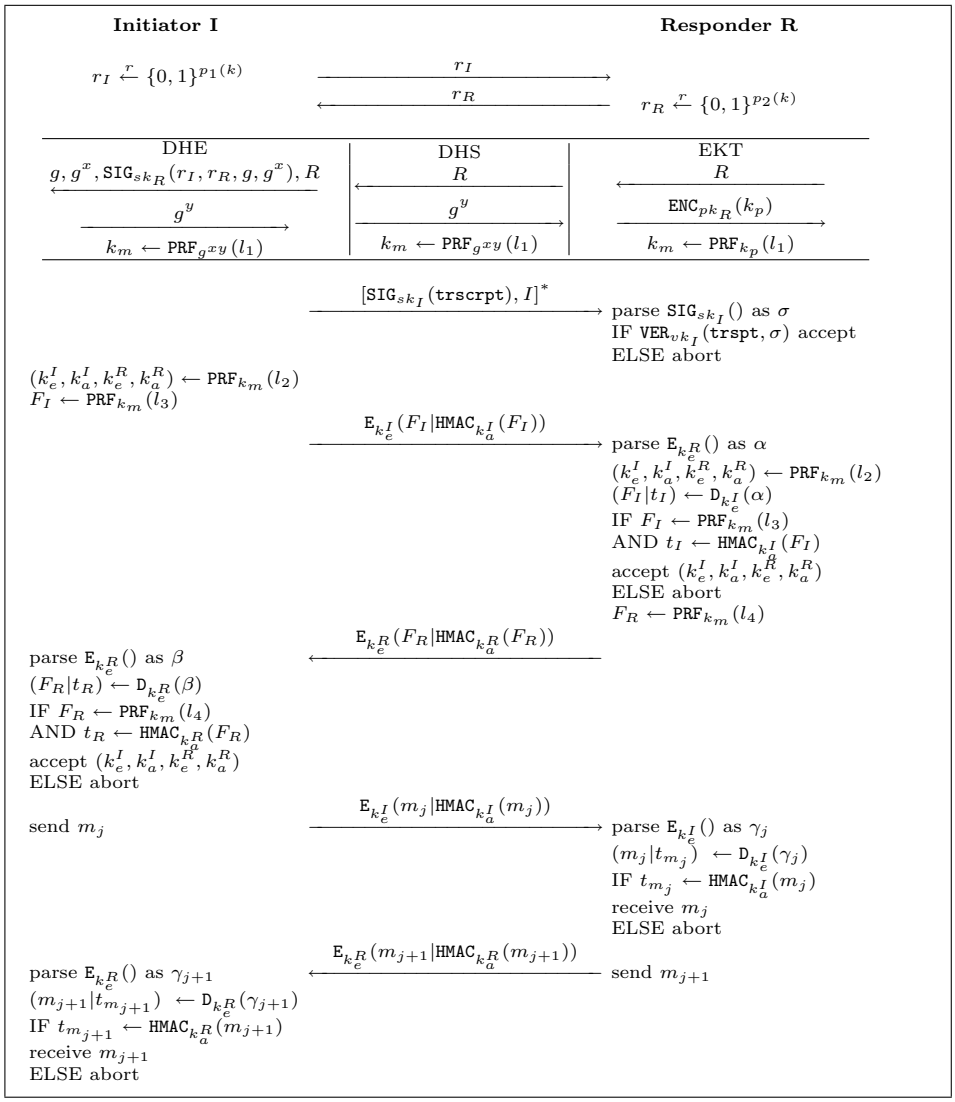

Fig. 1. The TLS protocol including the different subroutines DHE, DHS, and EKT to establish the master secret k_m. $[\cdot]^*$ marks the optional client authentication message. Event 'abort' invokes the alert protocol with the respective error message; events 'send' and 'receive' trigger interfaces to the application layer.

Handshake and Record Layer The TLS handshake aims at the negotiation of a common secret called the *master secret* k_m which is in turn derived from the the previously established *premaster secret* k_p. The modularity of the handshake protocol is captured by the fact that different subroutines are applied to establish the premaster secret and derive the master secret while the remaining structure of the handshake is unchanged (see Fig. 1). TLS distinguishes among the following subroutines: encryption of the premaster secret using the server's public key (EKT); static (DHS) or ephemeral signed (DHE) Diffie-Hellman key exchange. Optionally, TLS allows for the client authentication via a signature over all received values `trscrpt` which can be verified using the public key with the client certificate. The master secret k_m is then used to derive up to four cryptographic keys for the record layer: two symmetric encryption keys k_e^P (including an initialization vector for the block-cipher based encryption), and two authentication keys k_a^P, where $P \in \{I, R\}$. Finally, client and server confirm the negotiated security parameters by exchanging their finished messages which are derived from k_m and protected via *authenticated encryption* by the record layer (i.e. MAC of the plaintext is used as input to the symmetric encryption). The same protection is then applied to the subsequent application data.

Remark 1. Note that an application message may be fragmented and compressed when processed by the record layer. Therefore, the record layer encodes sequence numbers into the fragments and maintains a counter in order to prevent disorder. Note also that a key feature of TLS is session resumption in order to reduce server-sided performance penalties. The client names an earlier session that it intends to continue; if the server agrees, the previous master secret is used with the new nonces to generate new key material for the record layer. Though not explicitly treated in our paper, it is easy to see that the security of the abbreviated handshake follows from our analysis of the full handshake.

3.2 Roadmap for the Modular Analysis of TLS

The structure of the TLS framework advocates its modular analysis. Intuitively, the handshake protocol captures the cryptographic task of key exchange and the composition with the record layer protocol emulates secure transfer of application messages. However, the straightforward idea to model the complete handshake protocol as ideal key exchange functionality in order to negotiate the session keys and compose it with the record layer protocol in order to realize a secure communication sessions functionality fails in general. The handshake protocol does *not* securely realize the ideal key exchange functionality since it uses the derived session keys to encrypt and authenticate finished messages. Thus, the environment can test the keys using the finished messages and tell the two worlds apart.

In our analysis we avoid this obstacle by devising a functionality $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ that emulates the handshake's subroutines to negotiate the master secret k_m (instead of a straight-line computation of the session keys). $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ captures the fact that two players receive a random key unless either player is corrupted. Next, we

demonstrate that the subroutines DHE, DHS, and EKT securely realize $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ (Section 4). Our analysis is focused on responder-only and mutual authenticated communication which are the authentication modes supported by TLS (apart from anonymous Diffie-Hellman suites). Since TLS operates in a setting where a Certificate Authority (CA) is required, we formalize the global setup assumption by formulating the real-world protocols in $\mathcal{F}$-hybrid models, utilizing the *certification functionality* $\mathcal{F}_{\mathrm{CERT}}$, *certified public key encryption functionality* $\mathcal{F}_{\mathrm{CPKE}}$, and *certificate authority functionality* $\mathcal{F}_{\mathrm{CA}}$, as presented in [26, 27].

The composition with these functionalities to a subroutine protocol is preserved by the *universal composition with joint state (JUC) theorem*, proposed in [28]. This operation is similar to universal composition with the exception that multiple instances of a protocol can have a joint state. It is useful in the case of key exchange when multiple subroutine protocol sessions have access to the same instance of functionalities $\mathcal{F}_{\mathrm{CERT}}$, $\mathcal{F}_{\mathrm{CPKE}}$, and $\mathcal{F}_{\mathrm{CA}}$ that use the same key for authenticating multiple messages (i.e. the signature, encryption, and deposited key is the joint state, respectively). Finally, we make use of the composition theorem and specify the TLS protocol in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model. We show that the reformulated TLS protocol securely realizes the ideal functionality for the secure communication sessions (Section 5).

4 Analysis of TLS Subroutines

We proceed with the specification of an ideal-world functionality which we henceforth call *universal key exchange* $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ that captures the requirements of the subroutines DHE, DHS, and EKT. The key exchange functionality $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ is illustrated in Fig. 2. It mimics the cryptographic task that the players I and R agree upon a shared secret μ which is indistinguishable from an independently chosen value of the same length as long as a party is uncorrupted. There is a large body of literature that covers ideal key exchange functionalities (e.g. [2, 11, 27]). $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ is similar to these functionalities except for:

First, the players authenticate in a post-specified fashion, i.e. the environment invokes players with the session identifier `SID` and optionally their own identity. A player learns its peer identity while executing the TLS protocol (captured by the fact that peer identities are given by the functionality and not in the setup). This is an essential difference of TLS to related protocols (e.g. SSH) where the players have already negotiated their public keys before the protocol start.

Second, the functionality defines a hybrid notion of authenticated key exchange. When the initiator is parameterized with an identity, i.e. $ID_I{=}I$, the functionality assures mutual authentication between the initiator and server. Then, the functionality randomly fixes the (master) key unless a party is corrupt. On the other hand, when the initiator is invoked with an anonymous identity, i.e. $ID_I{=}\perp$, the functionality guarantees a matching conversation between the responder and some party whose identity is unknown. Consequently, the adver-

Functionality $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$

$\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ proceeds as follows when parameterized with security parameter k.

- Upon receiving an input ("establish-key", SID, ID_I) from some party, where $ID_I \in (\perp, I)$, record ID_I as initiator, and send a message ("establish-session", SID, ID_I) to the adversary. Upon receiving input ("establish-key", SID, R) from some other party, record R as responder, and send the message ("establish-key", SID, R) to the adversary.
- Upon receiving an answer ("impersonate", SID, $\tilde{\mu}$) from the adversary, do: If $ID_I{=}\perp$, record the adversary as initiator and send message ("Key", SID, $\perp$, $\tilde{\mu}$) to the responder. Else, ignore the message.
- Upon receiving an answer ("Key", SID, P, $\tilde{\mu}$) from the adversary, where P is either the initiator or responder, do: If neither initiator nor responder is corrupted, and there is no recorded key, fix μ uniformly from $\{0,1\}^k$. If either initiator or responder is corrupted, and there is no recorded key, record $\mu \leftarrow \tilde{\mu}$ as the adversary. Send message ("Key", SID, $\bar{P}$, μ) to P.

Fig. 2. The Universal Key Exchange Functionality. $\mathcal{F}_{\mathrm{KE}}^{2}$ is identical to $\mathcal{F}_{\mathrm{KE}}^{1}$ except that it excludes the impersonation query.

sary can impersonate the initiator and fix the master key.[4] The corresponding case in the real world is that the environment instructs the adversary to replay the key exchange protocol with the exception that it contributes to the pre-master key. The initiator is unable to terminate the session while the responder accepts the session. Technically, the functionality deploys the session identifier SID to determine the anonymous player. Such technicality is only feasible for a two party functionality. Recall that the SIDs of all Turing machines in a protocol instance must be identical in the UC framework. Any player participating in the same session who is not a responder must be a potential initiator.

Third, the functionality is defined for non-adaptively corrupting adversaries and therefore excludes (perfect) forward secrecy. In fact, this exclusion is precisely what makes it possible to define a single universal key exchange functionality which covers both, key transport and Diffie-Hellman key agreement.

Theorem 1. *Protocol* EKT *in the* $\mathcal{F}_{\mathrm{CPKE}}$, DHE *in the* $\mathcal{F}_{\mathrm{CERT}}$, *and* DHS *in the* $\mathcal{F}_{\mathrm{CA}}$*-hybrid model securely realize* $\mathcal{F}_{\mathrm{KE}}^{1}$. *Protocol* EKT *in the* ($\mathcal{F}_{\mathrm{CPKE}}$,$\mathcal{F}_{\mathrm{CERT}}$), DHE *in the* ($\mathcal{F}_{\mathrm{CERT}}, \mathcal{F}_{\mathrm{CERT}}$), *and* DHS *in the* ($\mathcal{F}_{\mathrm{CA}}, \mathcal{F}_{\mathrm{CERT}}$)*-hybrid model securely realize* $\mathcal{F}_{\mathrm{KE}}^{2}$.

The proof appears in the full version.

[4] Note that in case of Diffie-Hellman the key exchange functionality does not consider key control issues (see [5]). However, this has no impact on the security of secure communication sessions because the impersonator learns the master key and thus derives the session keys for the protection of the messages.

5 TLS UC-Realizes Secure Communication Sessions

The natural abstraction of TLS is to allow secure communication between players in a single protocol instance. While the handshake protocol aims at securely sharing uniformly distributed session keys, the record layer protocol provides authenticated encryption of session messages.

5.1 Universal Secure Communication Sessions

Secure communication sessions have been discussed in [2, 11] for the general case in which all players are authenticated. We refine the functionality and relax the requirements to the universal model of authentication in the post-specified setting, where a player learns the identity of its peer during the execution of the protocol and must cope with impersonation attacks against the initiator, provided the environment keeps the initiator's identity secret. In which case, we have to expect a real-world adversary that plays the role of the initiator by intercepting the first two protocol rounds, choosing own premaster secret, and completing the protocol in the normal way. The initiator will be unable to terminate the session. Nevertheless, the responder accepts the session and answers to the adversary, mimicking arbitrary party. We capture the requirements by formulating a universal secure communication sessions functionality $\mathcal{F}_{\text{SCS}}^{(1,2)}$ in Fig. 3. Let us highlight some characteristics of $\mathcal{F}_{\text{SCS}}^{(1,2)}$ in the following:

Functionality $\mathcal{F}_{\text{SCS}}^{(1,2)}$

$\mathcal{F}_{\text{SCS}}^{(1,2)}$ proceeds as follows, when parameterized by a leakage function $l : \{0,1\}^* \to \{0,1\}^*$.

- Upon receiving an input ("establish-session", `SID`, ID_I) from some party, where $ID_I \in (\perp, I)$, record ID_I as initiator, and send the message to the adversary. Upon receiving input ("establish-session", `SID`, R) from some party, record R as responder, and forward the message to the adversary.
- Upon receiving a value ("impersonate", `SID`) from the adversary, do:If (ID_I=$\perp$), check that no `ready` entry exists, and record the adversary as initiator. Else ignore the message.
- Upon receiving a value ("send", `SID`, m, $\bar{P}$) from party P, which is either initiator or responder, check that a record (`SID`, P, $\bar{P}$) exists, record `ready` (if there is no such entry) and send ("sent", `SID`, $l(m)$) to the adversary and a private delayed value ("receive", `SID`, m, P) to $\bar{P}$. Else ignore the message. If the sender is corrupted, then disclose m to the adversary. Next, if the adversary provides m' and no output has been written to the receiver, then send ("send", `SID`, m', P') to the receiver unless P' is an identity of an uncorrupted party.

Fig. 3. The Universal Secure Communication Sessions Functionality

First, the functionality handles a uni- and bi-directional model of authentication (as in the universal key exchange functionality). The latter is accomplished by invoking the players with their own identity. The first is realized by invoking the initiator with an empty identity value $\perp$ allowing the adversary to mount an impersonation attack. The functionality proceeds in the usual way except that a secure session is established between the adversary and the responder.

Second, the functionality guarantees that the adversary gains no information other than some side channel information about the transmitted plaintext m, expressed via a leakage function $l(m)$, when the adversary has neither impersonated nor corrupted a player. In particular, the information leakage includes the length and sequence number of m and some information concerning the transmitted messages' source and destination; thus, modeling network information about the TLS-protected channel from lower-layer protocols and higher-layer protocols prior to their processing by the record layer. (We remark that the environment may provide additional leakage information such as the domain name and the name of the client application. This leakage information may be important upon composition with a dedicated higher-layer protocol).

Third, the session identifier `SID` assures that the functionality may address the initiator even though its identity is undisclosed (because it knows the responder's identity and the underlying system model permits a party, i.e. the initiator, to interact with the functionality with an identical session identifier). This is so because TLS runs above transport-layer protocols which provide the players with a globally unique address (e.g. IP address). Furthermore, these protocols ensure that the channel is locally fresh by exchanging a pair of nonces.

Last, the functionality manages an internal ready state. This technicality ensures that in the responder-only model of authentication the adversary cannot impersonate the initiator after the responder agreed upon the session keys and switched into the pending state waiting for the transmission.

5.2 Protocol ρ realizes $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$

In Fig. 4 we apply Theorem 1 and reformulate protocol ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model. The general Universal Composability theorem guarantees that no probabilistic polynomial time-bounded environment distinguishes between the case that it observes an instance of TLS executing the subroutines DHE, DHS and EKT and the case that it interacts with a TLS instance where the subroutines are replaced by the ideal key exchange functionality. We are now ready to state our main theorem.

Theorem 2. *Protocol ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model securely realizes $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$.*

Proof. Let $\mathcal{A}$ be a real-world adversary that operates against ρ. We construct an ideal-world adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between the case that it interacts with $\mathcal{A}$ and parties running ρ in the $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$-hybrid model or with $\mathcal{S}$ in the ideal world for $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and mimics

Protocol ρ

1. Upon activation with query ("establish-session", SID, ID_I) by $\mathcal{Z}$, where $ID_I \in (\perp, I)$, the initiator sends the init message (r_I) where $r_I \xleftarrow{r} \{0,1\}^{p_1(k)}$ is a nonce. Upon activation with query ("establish-key", SID, R) by $\mathcal{Z}$, the responder waits for the receipt of the init message. It responds with own nonce $r_R \xleftarrow{r} \{0,1\}^{p_2(k)}$ and initializes a copy of $\mathcal{F}_{\text{KE}}^{(1,2)}$ with session identifier $\text{SID}_{\text{KE}}=(r_I|r_R)$ by sending query ("establish-key", SID_{KE}, R) to $\mathcal{F}_{\text{KE}}^{(1,2)}$.
2. Upon receiving the response message, the initiator calls $\mathcal{F}_{\text{KE}}^{(1,2)}$ with session identifier $\text{SID}_{\text{KE}}=(r_I|r_R)$ on query ("establish-key", SID_{KE}, ID_I) and waits for the delivery of output ("Key", SID_{KE}, R, μ). It then computes the session keys $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \text{PRF}_\mu(l_2)$ and the finished value $F_I \leftarrow \text{PRF}_\mu(l_3)$. Additionally, the initiator sends the final initiator message $(\text{E}_{k_e^I}(F_I|\text{HMAC}_{k_a^I}(F_I)))$.
3. When the responder receives the final initiator message (α), it first waits for the delivery of ("Key", SID_{KE}, ID_I, μ) from $\mathcal{F}_{\text{KE}}^{(1,2)}$. Then, the responder computes in the same way the session keys $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \text{PRF}_\mu(l_2)$ for the players. It decrypts the final initiator message $(F_I|t_I) \leftarrow \text{D}_{k_e^I}(\alpha)$ and verifies that $F_I \leftarrow \text{PRF}_\mu(l_3)$ and $t_I \leftarrow \text{HMAC}_{k_a^I}(F_I)$. If the verification fails, it aborts. Otherwise, it computes the finished value $F_R \leftarrow \text{PRF}_\mu(l_4)$ and sends the final responder message $(\text{E}_{k_e^R}(F_R|\text{HMAC}_{k_a^R}(F_R)))$.
4. Upon delivery of the final responder message (β), the initiator decrypts the message $(F_R|t_R) \leftarrow \text{D}_{k_e^R}(\beta)$. Then, it verifies that $F_R \leftarrow \text{PRF}_\mu(l_4)$ and $t_R \leftarrow \text{HMAC}_{k_a^R}(F_R)$. If the verification fails, it aborts.
5. Once the session keys are agreed upon, the sender $P \in (I, R)$ waits for the transmission notification ("send", SID, m, $\bar{P}$) from $\mathcal{Z}$. It then sends $\text{E}_{k_e^P}(m|t_m)$ whereby message m is authenticated through the tag $t_m \leftarrow \text{HMAC}_{k_a^P}(m)$. Upon receiving the message γ, the receiver $\bar{P}$ decrypts the message $(m|t_m) \leftarrow \text{D}_{k_e^P}(\gamma)$ and verifies that $t_m \leftarrow \text{HMAC}_{k_a^P}(m)$. If the verification fails, it aborts. Otherwise, the receiver accepts the message and makes the local output ("receive", SID, m, P) to $\mathcal{Z}$.

Fig. 4. The full TLS Framework Structure, in the $\mathcal{F}_{\text{KE}}^{(1,2)}$-hybrid Model

an interaction with players executing ρ. It tries to make the internal protocol simulation consistent with the real protocol execution and the limitation that it has no information about the transmitted message m other than its length $l(m)$. The simulator allows the adversary $\mathcal{A}$ to attack the simulated protocol execution in arbitrary way throughout the simulation. $\mathcal{S}$ emulates the protocol execution in such a way that $\mathcal{A}$ thinks that it intercepts a real-world execution of ρ, and such that its interaction with $\mathcal{Z}$ is distributed computationally indistinguishable from that observed by the environment in the real-world execution.

In detail, the simulator proceeds in the following way:

1. **Simulating invocation of I.** Upon receiving ("establish-session", SID, ID_I) from $\mathcal{F}_{\text{SCS}}^{(1,2)}$, $\mathcal{S}$ feeds $\mathcal{A}$ with the init message (r_I) where $r_I \xleftarrow{r} \{0,1\}^{p_1(k)}$.
2. **Simulating invocation of R.** Upon receiving ("establish-session", SID, R) from $\mathcal{F}_{\text{SCS}}^{(1,2)}$, $\mathcal{S}$ waits for receipt of an init message (r_I') from $\mathcal{A}$. Then, it

chooses a nonce $r_R \xleftarrow{r} \{0,1\}^{p_2(k)}$ and feeds $\mathcal{A}$ with the response message (r_R, R). Finally, it calls $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ on query ("establish-key", $\mathtt{SID}'_{\mathrm{KE}}$, R), where $\mathtt{SID}'_{\mathrm{KE}}$=$(\mathtt{SID} \circ \mathtt{r}'_{\mathtt{I}} | \mathtt{r}_{\mathtt{R}})$.

3. **Simulating receipt of a response message by I.** Upon $\mathcal{A}$ delivers the message (r'_R, P') to I, $\mathcal{S}$ proceeds as follows:
   (a) $\mathcal{S}$ verifies that I has previously sent the init message (r_I).
   (b) $\mathcal{S}$ checks that P'=R. Otherwise, it aborts the simulation.
   (c) $\mathcal{S}$ mimics on behalf of I the master key generation by invoking a copy of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$. The master key is obtained by handing $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ the message ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I), where $\mathtt{SID}_{\mathrm{KE}}$=$(\mathtt{SID} \circ \mathtt{r}_{\mathtt{I}} | \mathtt{r}'_{\mathtt{R}})$ and waiting for the delivery of the response message ("Key", $\mathtt{SID}_{\mathrm{KE}}$, R, μ). Otherwise, $\mathcal{S}$ terminates with an internal error message (because there was no matching activation of the same instance of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$ in form of a query ("establish-key", $\mathtt{SID}_{\mathrm{KE}}$, R) by the simulator on behalf of the responder).
   (d) $\mathcal{S}$ defines the master key μ, the session keys $(k_e^I, k_a^I, k_e^R, k_a^R)$, and the finished value F_I to be random values Δ_{k_m}, $(\Delta_{k_e^I}, \Delta_{k_a^I}, \Delta_{k_e^R}, \Delta_{k_a^R})$, and Δ_{F_I} chosen from the appropriate spaces, respectively.
   (e) $\mathcal{S}$ feeds $\mathcal{A}$ with the final initiator message $(\mathtt{E}_{\Delta_{k_e^I}}(\Delta_{F_I} | t_I))$, where $t_I \leftarrow \mathtt{HMAC}_{\Delta_{k_a^I}}(\Delta_{F_I})$.
4. **Simulating receipt of a final initiator message by R.** When $\mathcal{A}$ delivers the message (α) to R, $\mathcal{S}$ proceeds as follows:
   (a) $\mathcal{S}$ verifies that it has previously received an init message (r'_I) and sent a response message (r_R, R).
   (b) $\mathcal{S}$ waits for the master key by mimicking the key establishment process of $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$. Now we distinguish between the following two distinct cases.
   Case 1 (no impersonation): If $\mathcal{S}$ receives an answer ("Key", $\mathtt{SID}_{\mathrm{KE}}$, ID_I, μ) from $\mathcal{F}_{\mathrm{KE}}^{(1,2)}$, then no impersonation attack has occurred. In this case $\mathcal{S}$ uses for k_m, $(k_e^I, k_a^I, k_e^R, k_a^R)$, and F_I exactly the same values Δ_{k_m}, $(\Delta_{k_e^I}, \Delta_{k_a^I}, \Delta_{k_e^R}, \Delta_{k_a^R})$, and Δ_{F_I} that it has chosen on behalf of the initiator before. Then, it waits for the delivery of the final initiator message and applies the session keys to decrypt $(F'_I | t'_I) \leftarrow \mathtt{D}_{\Delta_{k_e^I}}(\alpha)$. $\mathcal{S}$ compares whether $F'_I = \Delta_{F_I}$ and $t'_I = t_I$. If the verification fails, it aborts the simulation. Otherwise, it chooses F_R to be a random value Δ_{F_R} from the same space and feeds $\mathcal{A}$ with the final responder message $(\mathtt{E}_{\Delta_{k_e^R}}(\Delta_{F_R} | t_R))$ where $t_R \leftarrow \mathtt{HMAC}_{\Delta_{k_a^R}}(\Delta_{F_R})$. Then, $\mathcal{S}$ prepares for the secure message exchange on behalf of R.
   Case 2 (impersonation): If $\mathcal{S}$ receives an answer ("Key", $\mathtt{SID}'_{\mathrm{KE}}$, P', $\tilde{\mu}$), then the original master key has been modified by the adversary implying the impersonation attack framing the initiator. In this case $\mathcal{S}$ computes $(k_e^I, k_a^I, k_e^R, k_a^R)$, and F_I as specified in the protocol, i.e. $(k_e^I, k_a^I, k_e^R, k_a^R) \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_2)$, and $F_I \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_3)$. Then, it waits for the delivery of the final initiator message and decrypts $(F'_I | t'_I) \leftarrow \mathtt{D}_{k_e^I}(\alpha)$. $\mathcal{S}$ compares whether $F'_I = F_I$ and $t'_I = \mathtt{HMAC}_{k_a^I}(F_I)$. If the verification fails, it aborts the simulation. Otherwise, $\mathcal{S}$ computes $F_R \leftarrow \mathtt{PRF}_{\tilde{\mu}}(l_4)$,

and feeds $\mathcal{A}$ with the final responder message $(\mathtt{E}_{k_e^R}(F_R|\mathtt{HMAC}_{k_a^R}(F_R)))$. Finally, $\mathcal{S}$ sends ("impersonate", SID) to $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$. This is exactly the point in the simulation where the adversary has impersonated the unauthenticated party. Then, $\mathcal{S}$ continues the simulation with the exception that the interaction proceeds with $\mathcal{A}$ and I aborts the protocol.

Note that in all subsequent simulation steps, $\mathcal{S}$ uses session keys (k_e^P, k_a^P) for $P \in (I, R)$ and finished values F_I and F_R obtained from one of the above two cases.

5. **Simulating receipt of a final responder message by I.** When $\mathcal{A}$ delivers the message (β) to an uncorrupted I, $\mathcal{S}$ proceeds as follows:
   (a) $\mathcal{S}$ verifies that it has previously sent an init message (r_I), received a response message (r'_R, P'), and sent a final initiator message $(\mathtt{E}_{\Delta_{k_e^I}}(\Delta_{F_I}|t_I))$.
   (b) $\mathcal{S}$ uses its own session keys $(\Delta_{k_e^R}, \Delta_{k_a^R})$ to decrypt β obtaining $F'_R|t'_R$. Since no responder impersonation attacks may occur it aborts the simulation if $F'_R \neq \Delta_{F_R}$ or $t'_R \neq t_R$ whereby Δ_{F_R} and t_R are the values used by $\mathcal{S}$ on behalf of R in the previous simulation step 4b (case 1). If the simulation does not abort then $\mathcal{S}$ prepares for the secure message exchange on behalf of I.
6. **Simulating Message Transmission.** Upon receiving ("sent", SID, $l(m)$) from $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$, $\mathcal{S}$ extracts from $l(m)$ the sender and receiver identities. It then chooses a random message $\Delta_m \xleftarrow{r} \{0,1\}^{l(m)}$ and feeds $\mathcal{A}$ with message $\mathtt{E}_{k_e^P}(\Delta_m|t_{\Delta_m})$ where $t_{\Delta_m} \leftarrow \mathtt{HMAC}_{k_a^P}(\Delta_m)$.
7. **Simulating Message Reception.** Upon receiving the message (γ), the receiver decrypts the message $(\Delta'_{m'}, t'_{\Delta_{m'}}) \leftarrow \mathtt{D}_{k_e^P}(\gamma)$ and then verifies that $t'_{\Delta_{m'}} \leftarrow \mathtt{HMAC}_{k_a^P}(\Delta_m)$ using its own keys. If the verification fails, it aborts. Otherwise, $\mathcal{S}$ signals $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ to send the message.
8. **Simulating Static Corruption** If one of the parties gets corrupted, then $\mathcal{S}$ proceeds by emulating a ρ protocol session, just as a honest party would play it. In particular, $\mathcal{S}$ uses the message m transmitted by $\mathcal{F}_{\mathrm{SCS}}^{(1,2)}$ in the emulation of the last protocol round.

The proof of indistinguishability to demonstrate the validity of $\mathcal{S}$ appears in the full version.

6 Conclusion

We have analyzed the TLS protocol family in the framework of Universal Composition. We have shown that the complete TLS protocol framework securely realizes secure communication sessions. Thus, future analysis of composite protocols can be considerably simplified by calling the secure communication functionality in the hybrid-model reformulation. The composition theorem preserves that security holds under general composition with arbitrary players. Our analysis is performed under the consideration of static corruptions, since this setting is suitable for the combined treatment of key transport *and* Diffie-Hellman protocol

suites specified within the TLS standard. A future work may include consideration of adaptive corruptions, and thus modeling of (perfect) forward secrecy, which seems to be achievable by the TLS protocol suites based on Diffie-Hellman but not key transport.

Acknowledgment

We would like to thank Dennis Hofheinz, Aggelos Kiayias, Ralf Küsters, and Ivan Visconti for fruitful discussions and their valuable feedback. The authors were supported by the European Commission (IST-2002-507932 ECRYPT).

References

1. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol, Version 1.1. RFC 4346, IETF (2006) Proposed Standard.
2. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS, IEEE Computer Society (2001) 136–145
3. Canetti, R., Fischlin, M.: Universally Composable Commitments. In Kilian, J., ed.: CRYPTO'01, LNCS 2139, pp. 19–40, Springer, 2001.
4. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally Composable Password-Based Key Exchange. In Cramer, R., ed.: EUROCRYPT'05, LNCS 3494, pp. 404–421, Springer, 2005.
5. Hofheinz, D., Müller-Quade, J., Steinwandt, R.: Initiator-Resilient Universally Composable Key Exchange. In Snekkenes, E., Gollmann, D., eds.: ESORICS'03, LNCS 2808, pp. 61–84, Springer, 2003.
6. Katz, J.: Universally Composable Multi-Party Computation Using Tamper-Proof Hardware. In Naor, M., ed.: EUROCRYPT'07, LNCS 4515, pp. 115–128, Springer, 2007.
7. Canetti, R., Krawczyk, H., Nielsen, J.: Relaxing Chosen-Ciphertext Security. Cryptology ePrint Archive, Report 2003/174, 2003.
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Secure Computation. In: STOC'02, pp. 494–503, ACM, 2002.
9. Kidron, D., Lindell, Y.: Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs. Cryptology ePrint Archive, Report 2007/478 (2007)
10. Canetti, R., Krawczyk, H.: Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In Yung, M., ed.: CRYPTO'02, LNCS 2442, pp. 143–161, Springer, 2002.
11. Canetti, R., Krawczyk, H.: Universally Composable Notions of Key Exchange and Secure Channels. In Knudsen, L.R., ed.: EUROCRYPT'02, LNCS 2332, pp. 337–351, Springer, 2002.
12. Schneier, B., Wagner, D.: Analysis of the SSL 3.0 Protocol. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, 1996
13. Bleichenbacher, D.: Chosen Ciphertext Attacks against Protocols based on the RSA Encryption Standard PKCS #1. In Krawczyk, H., ed.: CRYPTO'98, LNCS 1462, pp. 1–12, Springer, 1998.

14. Jonsson, J., Kaliski, B.: On the Security of RSA Encryption in TLS. In Yung, M., ed.: CRYPTO'02, LNCS 2442, pp. 127–142, Springer, 2002.
15. Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure is SSL?). In Kilian, J., ed.: CRYPTO'00, LNCS 2139, pp. 310–331, Springer, 2000.
16. Paulson, L.C.: Inductive Analysis of the Internet Protocol TLS. ACM Transactions on Computer and System Security **2**(3) (1999) 332–351
17. Mitchell, J.C., Shmatikov, V., Stern, U.: Finite-State Analysis of SSL 3.0. In: Proceedings of the 7th Conference on USENIX Security Symposium, pp. 16-16, 1998.
18. Ogata, K., Futatsugi, K.: Equational Approach to Formal Analysis of TLS. In: ICDCS'05, pp. 795–804, IEEE Computer Society, 2005.
19. He, C., Sundararajan, M., Datta, A., Derek, A., Mitchell, J.C.: A Modular Correctness Proof of IEEE 802.11i and TLS. ACM Conference on Computer and Communications Security (CCS'05), pp. 2–15, ACM, 2005.
20. Dolev, D., Yao, A.C.C.: On the Security of Public Key Protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–207
21. Morrissey, P., N.P.Smart, Warinschi, B.: A Modular Security Analysis of the TLS Handshake Protocol. Cryptology ePrint Archive, Report 2008/236, 2008.
22. Jonsson, J.: Security Proofs for the RSA-PSS Signature Scheme and Its Variants. Cryptology ePrint Archive, Report 2001/053, 2001.
23. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: CRYPTO'96, LNCS 1109, pp. 1–15, Springer, 1996.
24. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Okamoto, T., ed.: ASIACRYPT'00, LNCS 1976, pp. 531–545, Springer, 2000.
25. Fouque, P.A., Pointcheval, D., Zimmer, S.: HMAC is a Randomness Extractor and Applications to TLS. In: AsiaCCS '08, pp. 21–32, ACM Press, 2008.
26. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW'04, pp. 219–233, IEEE CS, 2004.) Full version at `http://eprint.iacr.org/2003/239`.
27. Canetti, R., Herzog, J.: Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In: TCC'06, LNCS 3876, pp. 380–403, Springer, 2006.
28. Canetti, R., Rabin, T.: Universal Composition with Joint State. In Boneh, D., ed.: CRYPTO'03, LNCS 2729, pp. 265–281, Springer, 2003.
29. Hansen, S., Skriver, J., Nielson, H.: Using Static Analysis to Validate the SAML Single Sign-On Protocol. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security, 2005.
30. Groß, T., Pfitzmann, B., Sadeghi, A.R.: Browser Model for Security Analysis of Browser-Based Protocols. In di Vimercati, S.D.C., Syverson, P.F., Gollmann, D., eds.: ESORICS'05, LNCS 3679, pp. 489–508, Springer, 2005.
31. Groß, T., Pfitzmann, B., Sadeghi, A.R.: Proving a WS-Federation Passive Requestor Profile with a Browser Model. In: Workshop on Secure Web Services, ACM Press, 2005.