A Standard-Model Security Analysis of TLS-DHE

Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk

Horst Görtz Institute for IT Security
Chair for Network- and Data Security
Ruhr-University Bochum, Germany
{tibor.jager,florian.kohlar,sven.schaege,joerg.schwenk}@rub.de

September 30, 2011

Abstract

TLS is the most important cryptographic protocol in use today. However, up to now there is no complete cryptographic security proof in the standard model, nor in any other model. We give the first such proof for the TLS ciphersuites based on ephemeral Diffie-Hellman key exchange (TLS-DHE), which include the cipher suite `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` mandatory in TLS 1.0 and TLS 1.1. Due to subtle problems with the encryption of the final `Finished` messages of the TLS handshake, this proof cannot be formulated in the Bellare-Rogaway (BR) or any other indistinguishability-based model. Therefore we only prove the security of a truncated version of the TLS handshake (which has been the subject of nearly all previous papers on TLS) completely in the standard BR model. We then define the notion of authenticated and confidential channel establishment (ACCE) as a model in which the combination of TLS handshake and TLS Record Layer can be proven secure.
Keywords: authenticated key agreement, SSL, TLS, provable security, ephemeral Diffie-Hellman

Contents

1 Introduction

1.1 TLS and Authenticated Key Agreement

Transport Layer Security (TLS) is the single most important Internet security mechanism today. Due to a subtle interleaving of the TLS handshake protocol with the TLS Record Layer it is impossible to prove the security of TLS using well-established security models [11, 20, 19] which define security via indistinguishability of keys. Therefore there is no security proof for the complete protocol up to now, as we illustrate below. Instead, all prior work either considered a modified version of the TLS handshake [36, 47], or weaker security goals [34].

In this paper we provide new security results for TLS. First we give a formal proof that the truncated version of the TLS handshake protocol, which has been subject to prior work on TLS [36, 47], is an authenticated key agreement protocol in the Bellare-Rogaway model. This enables direct comparison of our results. Then we extend both the model and the proof to cover the combination of TLS handshake and TLS Record Layer, which allows us to show the security of *full* TLS ciphersuites. We consider TLS ciphersuites based on *ephemeral Diffie-Hellman* key agreement (TLS-DHE), and assume that the building blocks of TLS (pseudo-random function, digital signature scheme, MAC, symmetric cipher) meet certain classical security properties. Our proof is stated for mutual authentication, i.e. the client authenticates itself using a client certificate. This allows us to base our work on standard definitions for secure authenticated key agreement (AKE) protocols. The result can be extended to server-only authentication. [1]

We were not able to extend our techniques to RSA encrypted key transport, which is the basis for the new mandatory ciphersuite in TLS 1.2. Such a result will be difficult to achieve, since it would imply the security of RSA encryption with PKCS#1 v1.5 padding in the standard model. In previous work this problem was circumvented by either using the Random Oracle Model [47], or by assuming PKCS#1 v2.0 (RSA-OEAP), which is not used in TLS, and omitting authentication [34].

Authenticated key agreement (AKE) is a basic building block in modern cryptography. Many secure protocols for two-party and group key agreement have been proposed, including generic compilers that transform simple key agreement protocols into authenticated key agreement protocols, with many additional security properties. However, since many different formal models for different purposes exist [9, 11, 15, 19, 20, 25, 43, 23], choice of the right model is not an easy task, and must be considered carefully. We have to take into account that we cannot modify any detail in the TLS protocol, nor in the network protocols preceding it.

We have chosen in essence the first model of Bellare and Rogaway [11] and the ability of the adversary to perform adaptive queries. Essentially equivalent variants of this model have been studied by [23, 15], and especially by [47]. The reasons for our choice are given in Appendix A.

Proving Security of TLS. Morissey *et al.* [47] consider a truncated handshake protocol, where the final encryption of the `Finished` messages is omitted. This modification is necessary, since the full TLS handshake does not provide indistinguishable keys due to an interleaving of the key exchange part of TLS (the TLS handshake protocol) and the data encryption in the TLS record layer. This interleaving provides a 'check value' that allows to test whether a given key is 'real' or 'random'. More precisely, the final messages of the TLS handshake protocol (the `Finished`

[1] Please note that TLS-DHE offers *perfect forward security*, in contrast to TLS-DH or TLS-RSA

messages), which are essential to provide security against active adversaries like e.g. man-in-the-middle attackers, are first prepended with constant byte values (which provides us with known plaintext), then MACed and encrypted with the keys obtained from the handshake protocol. Thus, whenever an adversary receives a challenge key in response to a Test query, he can try to decrypt the `Finished` message and check validity of the *message authentication code* (MAC). If this succeeds, he will output 'real', and otherwise 'random'. Even changing the behavior of the Test query to only return the decryption keys (and not the MAC keys) does not help, since the adversary could still use the known plaintext bytes to answer the Test query successfully. Therefore it is impossible to prove the full TLS handshake protocol secure in any security model based on indistinguishability of *keys*.

1.2 Contribution

The paradox that the most important AKE protocol cannot be proven secure in any existing security model can be solved in two ways. Either one considers a truncated version of the TLS handshake by omitting the encryption of the two `Finished` messages, or a new model for the combination of the TLS-DHE handshake protocol and the record layer protocol must be devised. In this paper we follow both approaches.

First we give a security proof for the truncated version of the TLS-DHE handshake protocol. This allows to compare our results with previous work. We show that the truncated TLS-DHE handshake is a secure AKE protocol in the standard Bellare-Rogaway model. We do not require random oracles [10] or similar idealizations as in [47]. The proof relies solely on the DDH assumption, and the assumption that the building blocks of TLS (i.e. the signature scheme and the pseudo-random function) have certain standard security properties. It remains to consider whether the building blocks have the required properties. Here we can build on previous works that analyzed particular TLS components. (See Section 1.3 for details.)

Second we define the notion of authenticated and confidential channel establishment (ACCE). ACCE protocols are an extension of AKE protocols, in the sense that the symmetric cipher is integrated into the model. Loosely speaking, an ACCE channel guarantees that a message written on this channel can only be read by a party X which has identity pk_X, and that all messages read from this channel indeed originate from party X. In contrast to AKE protocols, where one requires *key indistinguishability*, we demand that a secure ACCE protocol allows to establish a "secure channel" in the sense of stateful length-hiding authenticated encryption [50]. This captures exactly the properties expected from TLS-like protocols in practice. We prove that *full* TLS-DHE ciphersuites, i.e., the combination of TLS Handshake and TLS Record Layer, forms a secure ACCE protocol, if the record layer protocol provides security in the sense of length-hiding authenticated encryption. Note that the latter was proven recently by Paterson et al. [50] for CBC-based record layer protocols.

1.3 Security Requirements on TLS Building Blocks

In our proofs we reduce the security of ephemeral Diffie-Hellman ciphersuites to certain security properties of building blocks of TLS. These building blocks are (see Section 3 for precise definitions):

PSEUDORANDOM FUNCTION. Our proofs require that the pseudo-random function PRF meets the standard security definition for pseudo-random functions, see Definition 3. All TLS versions specify a construction of PRF from cryptographic hash functions. TLS 1.2 prescribes SHA-256 [30], while previous standards used MD5 [53] and SHA-1 [31].

When using an ephemeral Diffie-Hellman based ciphersuite, there is a special requirement on the pseudo-random function in use. Depending on the state of the protocol, the PRF is seeded with keys from two different input spaces, namely either random bitstrings, or random elements of a prime-order group (either a group defined over an elliptic curve or a *subgroup* of Z_p^*). Fortunately, Fouque *et al.* [33] where able to show that the pseudo-random function of TLS 1.2 indeed constitutes a secure randomness extractor for *both* input spaces simultaneously (albeit under different security assumptions, which however all are related to the fact that the compression function of the underlying hash function behaves like a pseudo-random function), and thus implements the required pseudo-random function. Their work focuses on TLS 1.2 while stressing that the implementation of the key derivation function is not very different from the previous standards. We believe that similar results can easily be obtained for TLS 1.0 and TLS 1.1.

Symmetric encryption. The purpose of the TLS protocol is to establish an authenticated symmetric secret between two parties first (in the TLS handshake), and then to use this secret to provide a "secure channel" based on symmetric encryption (in the TLS record layer). While the informal idea of a "secure channel" is simple, defining its security requirements precisely is not so trivial.

For instance, it is well-known that using e.g. IND-CCA secure encryption in the record layer is not sufficient to provide what is expected from a secure TLS channel, since it does not prevent many relevant attacks. For instance, it does not rule out replaying, dropping or re-ordering of ciphertexts. This is certainly not desirable in a "secure channel", since it may lead to various kinds of attacks (cf. [18]). This issue can be solved by using a suitable *stateful* encryption scheme [5, 6]. For instance, TLS uses a "MAC-then-Encode-then-Encrypt" (MEE) approach where a sequence counter is included in the MAC of each ciphertext. Moreover, it is well-known that sometimes even only the plaintext *length* may reveal valuable informations to an adversary, such as web browsing habits (e.g. [56]) or spoken phrases in Voice-over-IP connections (e.g. [60]). Therefore TLS may utilize variable-length encoding to conceal the plaintext length up to some granularity.

To capture such requirements, the notion of *stateful length-hiding authenticated encryption* (stateful LHAE) was recently introduced by Paterson et al. [50], as a formalization of the security properties that are expected from the record layer of TLS. The authors of [50] were able to show that CBC-based record layer protocols of TLS 1.1 and 1.2 provably meet this security goal under reasonable assumptions.[2] The results are not applicable to TLS 1.0, since it is well-known that this version is insecure against chosen-plaintext attacks [2, 3] since intialization vectors are not always chosen at random.

Digital signature. Our analysis furthermore requires that the employed signature scheme is secure against existential forgeries under adaptive attacks (Definition 2). The current TLS standards offer three different signature schemes for authentication: RSASSA-PKCS#1 v1.5 [37], DSA [44], and ECDSA [35]. To our knowledge there exists currently no security proof for these signature schemes (under standard complexity assumptions). In the random oracle model, DSA and ECDSA are provably secure. More details can be found in [52, 57].

[2]The proceedings version of [50] contains only a proof of *stateless* LHAE security. However, as also noted in [50], it is straightforward to adopt the results to the stateful setting. It was also announced that a full proof will appear in the full version of [50].

Interpretation. Our results show that the TLS protocol framework itself is cryptographically sound, if the building blocks are suitably secure. By combining our work with [50] we obtain a standard-model security proof of TLS 1.1 and 1.2 for *current* ciphersuites if we assume directly that the signature scheme is EUF-CMA secure.[3]

Our results can also be seen as a 'stepping stone' towards a TLS version with a *complete* security proof in the standard model. Essentially, we identify certain security properties and prove that the TLS protocol framework yields a secure ACCE protocol under the assumption that the TLS building blocks satisfy these properties. Thus, in order to obtain a TLS version with complete security proof in the standard model, it suffices to specify a ciphersuite consisting of suitable building blocks that have the desired security property. This seems achievable, since all requried security notions are cryptographic *standard* definitions that have been shown to be achievable with practical cryptosystems. In particular, we can build upon the results of Foque *et al.* [33] on the security of the PRF, the results of Paterson *et al.* [50] on the security of the record layer protocols, and a large number of practical signature schemes with provable EUF-CMA security in the standard model, e.g. [32, 59, 17].

The proof that the truncated TLS handshake is a secure AKE protocol may be extended to static DH, since this only involves a valid simulation for the DH game in Lemma 1. For RSA encrypted key transport, the situation is more complex: Besides the know weaknesses of the PKCS#1 padding scheme, a valid proof must take into account that the server only authenticates itself with the `ServerFinished` message (m_{13} in Fig. 2), whereas our proof assumes that this happens already with message m_3 or m_4. Thus the structure of such a proof may be completely different. In previous publications, this problem was circumvented by either using a highly non-standard model [47], or by omitting authentication [34].

1.4 Related Work

Because of its eminent role, TLS and its building blocks have been subject to several security analyses. In 1996, Schneier and Wagner presented several minor flaws and some new active attacks against SSL 3.0 [58]. Starting with the famous Bleichenbacher attack [16], many papers focus on various versions of the PKCS#1 standard [37] that defines the encryption padding used in TLS with RSA-encrypted key transport [24, 36, 39, 38]. At Crypto'02, Johnson and Kaliski showed that a simplified version of TLS with padded RSA is IND-CCA secure when modeling TLS as a 'tagged key-encapsulation mechanism' (TKEM) [36] under the strong non-standard assumption that a 'partial RSA decision oracle' is available.

In an independent line of research, several works analyzed (simplified versions of) TLS using automated proof techniques in the Dolev-Yao model [29]. Proofs that rely on the Dolev-Yao model view cryptographic operations as deterministic operations on abstract algebras. There has been some work on simplified TLS following the theorem proving and model checking approach, i.e. Mitchell *et al.* used a finite-state enumeration tool named Murphi [46] while Ogata and Futatsugi used the interactive theorem prover OTS/CafeObj [49]. Paulson used the inductive method and the theorem prover Isabelle [51]. Unfortunately it is not known if these proofs are actually cryptographically sound.

Bhargavan *et al.* [14] go two steps farther: First, they automatically derive their formal model

[3]To our best knowledge there is no proof for this for the currently used schemes, but also no result contradicting these assumptions.

from the source code of an TLS implementation, and second they try to automatize computational proofs using the CryptoVerif tool. However, the results are rather trivial. Chaki and Datta [22] also use source code of TLS, automatically find a weakness in OpenSSL 0.9.6c, and claim that SSL 3.0 is correct.

In 2008, Gajek *et al.* presented the first security analysis of the complete TLS protocol, combining handshake protocol and Record Layer, in the Universal Composability framework [19] for all three key agreement protocols static Diffie-Hellman, ephemeral signed Diffie-Hellman, and encrypted key transport [34]. The nonces r_C and r_S exchanged between client and server can be seen as an instantiation of the protocol of Barak *et al.* [1] to agree on a globally unique session id. However, the ideal functionalities described in this paper are strictly weaker than the security guarantees we expect from TLS: For the handshake part, only unauthenticated key agreement is modelled ($\mathcal{F}_{KE}$), and thus the secure channel functionality ($\mathcal{F}_{SCS}$) only guarantees confidentiality, not authenticity of endpoints. The paper further assumes that RSA-OEAP is used for encrypted key transport, which is not the case for current versions of TLS.

Küsters and Tuengerthal [42] claim to prove composable security for TLS assuming only local session identifiers, but leave out all details of the proof and only point to [34].

Morissey *et al.* [47] analyzed, in a paper that is closest to our results, the security of the truncated TLS handshake protocol (cf. Section 5) in the random oracle model and provided a modular proof of security for the established application keys. They make extensive use of the random oracle model to separate the three layers they define in the TLS handshake, and to switch from computational to indistinguishability based security models. The proof of Morissey *et al.* proceeds in three steps. They first consider a very weak class of passively secure key agreement protocols where the session key cannot *be computed* from the session transcript. As an example, when considering encrypted key transport (of the premaster secret) this requirement can easily be fulfilled if the employed public key encryption scheme is OW-CPA secure. Next they define a slightly stronger security notion that additionally protects against unknown key share attacks and show that it applies to the master secret key agreement of TLS. As before security of the key is defined in a one-way sense. In the last step they show that the 'application keys' (i.e. the encryption keys and MAC keys) produced by TLS fulfill the standard notion of security, namely *indistinguishability* from random values. The use of the random oracle model is justified by the authors by the fact that it seems impossible to prove the PKCS#1 v1.5 based ciphersuites of TLS secure in the standard model. This argumentation does not affect our work, since we only consider Diffie-Hellman-based ciphersuites.

In a very recent work, Paterson, Ristenpart, and Shrimpton [50] introduce the notion of length-hiding authenticated encryption, which aims to capture the properties from the TLS record layer protocols. Most importantly, they were able to show that CBC-based ciphersuites of TLS 1.1 and 1.2 meet this security notion. This work matches nicely our results on the TLS handshake protocol. Their paper extends the seminal work of Bellare and Namprempre [7, 8] on authenticated encryption, and on the analysis of different Mac-then-Encode-then-Encrypt (MEE) schemes analyzed by Krawczyk [40] and Maurer and Tackmann [45].

2 Preliminaries and Definitions

In this section, we recall the required definitions for our result on the TLS protocol. We denote with $\emptyset$ the empty string, and with $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ the set of integers between 1 and n. If A

is a set, then $a \xleftarrow{\$} A$ denotes the action of sampling a uniformly random element from A. If A is a probabilistic algorithm, then $a \xleftarrow{\$} A$ denotes that A is run with fresh random coins and returns a.

2.1 The Decisional Diffie-Hellman Assumption

Let G be a group of prime order q. Let g be a generator of G. Given, (g, g^a, g^b, g^c) for $a, b, c \in \mathbb{Z}_q$ the decisional Diffie-Hellman (DDH) assumption says that it is hard to decide whether $c = ab \bmod q$.

Definition 1. We say that the DDH problem is *$(t, \epsilon_{\mathsf{DDH}})$-hard* in G, if for all adversaries $\mathcal{A}$ that run in time t it holds that

$$\left|\Pr\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \Pr\left[\mathcal{A}(g, g^a, g^b, g^c) = 1\right]\right| \leq \epsilon_{\mathsf{DDH}},$$

where $a, b, c \xleftarrow{\$} \mathbb{Z}_q$.

2.2 Digital Signature Schemes

A digital signature scheme is a triple $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$, consisting of a key generation algorithm $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$ generating a (public) verification key pk and a secret signing key sk on input of security parameter κ, signing algorithm $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(sk, m)$ generating a signature for message m, and verification algorithm $\mathsf{SIG.Vfy}(pk, \sigma, m)$ returning 1, if σ is a valid signature for m under key pk, and 0 otherwise.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger generates a public/secret key pair $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$, the adversary receives pk as input.

2. The adversary may query arbitrary messages m_i to the challenger. The challenger replies to each query with a signature $\sigma_i = \mathsf{SIG.Sign}(sk, m_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some $q \in \mathbb{N}$. Queries can be made adaptively.

3. Eventually, the adversary outputs a message/signature pair (m, σ).

Definition 2. We say that SIG is *$(t, \epsilon_{\mathsf{SIG}})$-secure* against *existential forgeries under adaptive chosen-message attacks* (EUF-CMA), if for all adversaries $\mathcal{A}$ that run in time t it holds that

$$\Pr\left[(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathcal{C}}(1^\kappa, pk) \text{ such that } \mathsf{SIG.Vfy}(pk, m, \sigma) = 1 \wedge m \notin \{m_1, \ldots, m_q\}\right] \leq \epsilon_{\mathsf{SIG}}.$$

Note that we have $q \leq t$, i.e. the number of allowed queries q is bound by the running time t of the adversary.

2.3 Pseudo-Random Functions

A *pseudo-random function* is an algorithm PRF. This algorithm implements a deterministic function $z = \mathsf{PRF}(k, x)$, taking as input a key $k \in \mathcal{K}_{\mathsf{PRF}}$ and some bit string x, and returning a string $z \in \{0, 1\}^\kappa$.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger samples $k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ uniformly random.

2. The adversary may query arbitrary values x_i to the challenger. The challenger replies to each query with $z_i = \mathsf{PRF}(k, x_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some $q \in \mathbb{N}$. Queries can be made adaptively.

3. Eventually, the adversary outputs value x and a special symbol $\top$. The challenger sets $z_0 = \mathsf{PRF}(k, x)$ and samples $z_1 \overset{\$}{\leftarrow} \{0,1\}^\kappa$ uniformly random. Then it tosses a coin $b \overset{\$}{\leftarrow} \{0,1\}$, and returns z_b to the adversary.

4. Finally, the adversary outputs a guess $b' \in \{0,1\}$.

Definition 3. We say that PRF is a *$(t, \epsilon_{\mathsf{PRF}})$-secure* pseudo-random function, if aa adversary running in time t has at most an advantage of ϵ_{PRF} to distinguish the PRF from a truly random function, i.e.

$$\left|\Pr\left[b = b'\right] - 1/2\right| \leq \epsilon_{\mathsf{PRF}}.$$

Again the number of allowed queries q is upper bounded by t (see Def. 2).

Remark 1. In 2008, Fouque *et al.* [33] showed that the HMAC-based key-derivation function of TLS is a pseudo-random function for 1) $\mathcal{K}_{PRF} = S$, where S is a prime-order group of size $|S| = q$ that is either defined over an elliptic curve or as a subgroup of $\mathbb{Z}_p^*$ such that $q|p-1$, and 2) $\mathcal{K}_{PRF} = \{0,1\}^l$ where l is the size of the master-secret ($l = 384$). The underlying security assumptions are all related to the fact that the compression function of the hash function used in HMAC behaves like a pseudo-random function. More details can be found in [33].

2.4 Stateful Length-Hiding Authenticated Encryption

Let us now describe the stateful variant of LHAE security (the following description and security model was obtained from the authors of [50] via personal communication).

A *stateful symmetric encryption scheme* consists of two algorithms $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$. Algorithm $(C, st_e') \overset{\$}{\leftarrow} \mathsf{StE.Enc}(k, \ell, H, m, st_e)$ takes as input a secret key $k \in \{0,1\}^\kappa$, an output ciphertext length $\ell \in \mathbb{N}$, some header data $H \in \{0,1\}^*$, a plaintext $m \in \{0,1\}^*$, and the current state $st_e \in \{0,1\}^*$, and outputs a ciphertext $C \in \{0,1\}^*$ and an updated state st_e'. Algorithm $(m', st_d') = \mathsf{StE.Dec}(k, H, C, st_d)$ takes as input a key k, header data H, a ciphertext C, and the current state $st_d \in \{0,1\}^*$, and returns an updated state st_d' and a value m' which is either the message encrypted in C, or a distinguished error symbol $\perp$ indicating that C is not a valid ciphertext. Both encryption state st_e and decryption state st_d are initialized to the empty string $\emptyset$. Algorithm $\mathsf{StE.Enc}$ may be probabilistic, while $\mathsf{StE.Dec}$ is always deterministic.

Definition 4. We say that a stateful symmetric encryption scheme $\mathsf{StE} = (\mathsf{StE.Init}, \mathsf{StE.Enc}, \mathsf{StE.Dec})$ is $(t, \epsilon_{\mathsf{sLHAE}})$-secure, if $\Pr[b = b'] \leq \epsilon_{\mathsf{sLHAE}}$ for all adversaries $\mathcal{A}$ running in time at most t in the following experiment.

- Choose $b \overset{\$}{\leftarrow} \{0,1\}$ and $k \overset{\$}{\leftarrow} \{0,1\}^\kappa$, and set $st_e := \emptyset$ and $st_d := \emptyset$,
- run $b' \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}$.

Here $\mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}$ denotes that $\mathcal{A}$ has access to two oracles $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$. The encryption oracle $\mathsf{Encrypt}(m_0, m_1, \ell, H)$ takes as input two messages m_0 and m_1. It maintains a counter i which is initialized to 0. Oracle $\mathsf{Decrypt}(C, H)$ takes as input a ciphertext C and header H, and keeps a counter j and a variable phase, both are initialized to 0. Both oracles proceed a query as defined in Figure 1.

Figure 1: Encrypt and Decrypt oracles in the stateful LHAE security experiment.

```
$\underline{\mathsf{Encrypt}(m_0, m_1, \ell, H)}$:
$i := i + 1$
$(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \ell, H, m_0, st_e)$
$(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \ell, H, m_1, st_e)$
If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return $\perp$
$(C_i, st_e) := (C^{(b)}, st_e^{(b)})$
Return $C_i$
```

```
$\underline{\mathsf{Decrypt}(C, H)}$:
$j := j + 1$
If $b = 0$, then return $\perp$
$(m, st_d) = \mathsf{StE.Dec}(k, H, C, st_d)$
If $j > i$ or $C \neq C_j$, then $\mathsf{phase} := 1$
If $\mathsf{phase} = 1$ then return $m$
Return $\perp$
```

3 Transport Layer Security

The current version of TLS is 1.2 [28] and coexists with its predecessors TLS 1.0 [26] and TLS 1.1 [27]. Figure 2 illustrates the TLS protocol with ephemeral Diffie-Hellman key exchange and client authentication (i.e. ciphersuites `TLS_DHE_*`). This figure and the following description are valid for *all* TLS versions since v1.0.

The TLS handshake protocol consists of 13 messages, whose content ranges from constant byte values to tuples of cryptographic values. Not all messages are relevant for our security proof, we list them merely for completeness. All messages are prepended with a numeric tag that identifies the type of message, a length value, and the version number of TLS. All messages are sent through the 'TLS Record Layer', which at startup provides no encryption nor any other cryptographic transformations.

Client hello. Message m_1 is the `Client Hello` message. It contains four values, two of which are optional. For our analysis the only important value is r_C, the random value chosen by the client. It consists of 32 bytes (256 Bits), where 4 Bytes are usually used to encode the local time of the client. The remaining 28 Bytes are chosen randomly by the client. If the client wants to resume a previous TLS session, he may optionally include a *TLS session ID* value received from the server in a previous session. (This value is not protected cryptographically and should thus not be confused with session IDs used in formal security proofs.) This is followed by a list `cs-list` of *ciphersuites*, where each ciphersuite is a tuple of key exchange method, signing, encryption and MAC algorithms, coded as two bytes. Data compression is possible before encryption and is signaled by the inclusion of zero or more compression methods.

Server hello. The `Server Hello` message m_2 has the same structure as `Client Hello`, with the only exception that at most one ciphersuite and one compression method can be present. In our analysis the random value r_S is important. The server may send a TLS session ID sID to the client. Message m_3 may contain a chain of certificates, starting from the TLS server certificate up to a direct child of a root certificate. Since we do not include public key infrastructures in our analysis (the identity of each party is its public key pk_S), one certificate $cert_S$ containing pk_S (which may be self-signed) is sufficient for this paper. The public key in the certificate must match the ciphersuite chosen by the server. For ephemeral Diffie-Hellman key exchange, the public key may be any key that can be used to sign messages. The Diffie-Hellman (DH) key exchange parameters are contained in the `Server Key Exchange` message m_4, including information on the DH group (e.g. prime number p and generator g for a prime-order q subgroup of $\mathbb{Z}_p^*$), the DH share T_S, and

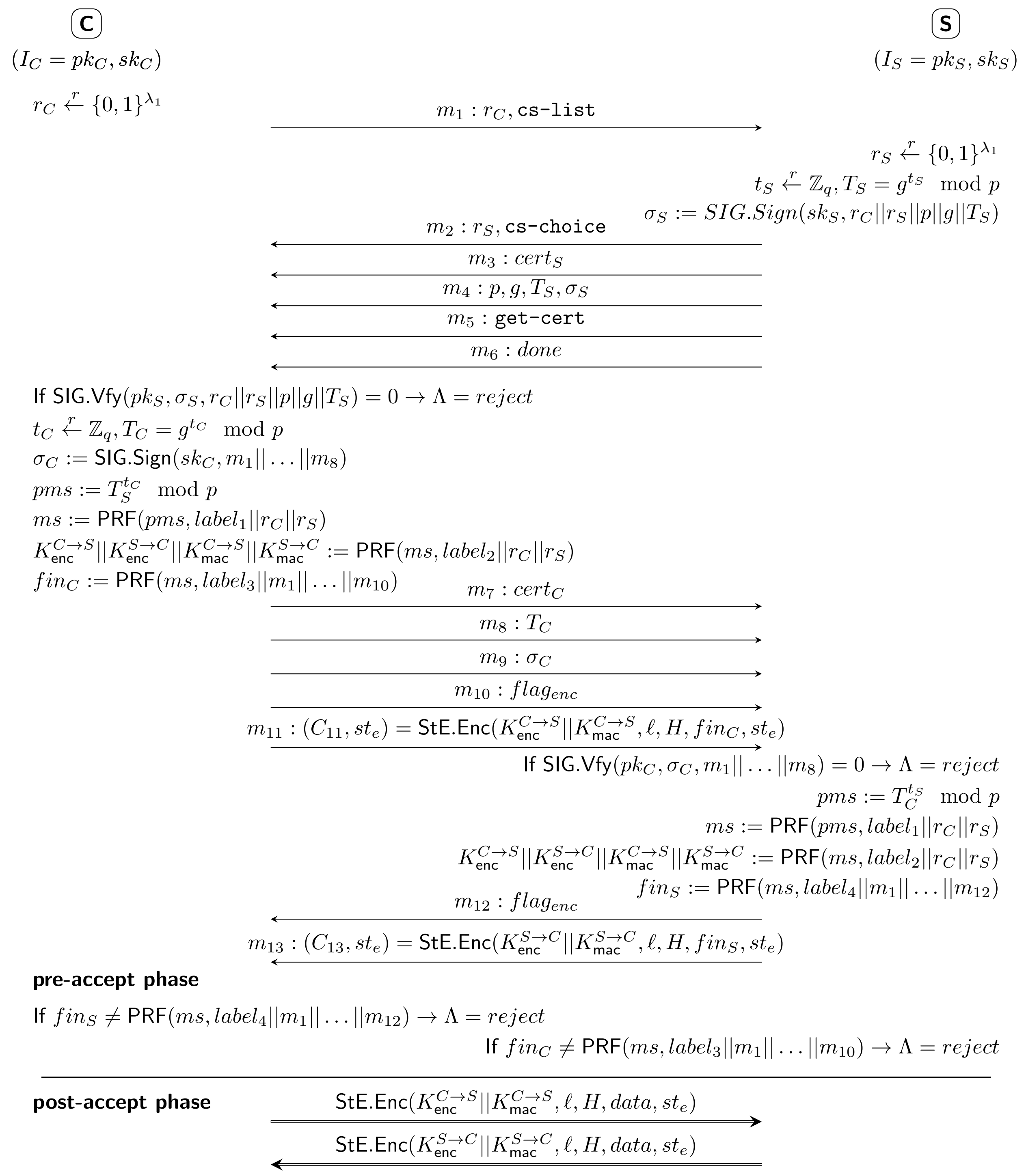
C
S
$(I_C = pk_C, sk_C)$
$(I_S = pk_S, sk_S)$
$r_C \xleftarrow{r} \{0,1\}^{\lambda_1}$
$m_1 : r_C$, `cs-list`
$r_S \xleftarrow{r} \{0,1\}^{\lambda_1}$
$t_S \xleftarrow{r} \mathbb{Z}_q, T_S = g^{t_S} \mod p$
$\sigma_S := SIG.Sign(sk_S, r_C||r_S||p||g||T_S)$
$m_2 : r_S$, `cs-choice`
$m_3 : cert_S$
$m_4 : p, g, T_S, \sigma_S$
m_5 : `get-cert`
$m_6 : done$
If $\mathsf{SIG.Vfy}(pk_S, \sigma_S, r_C||r_S||p||g||T_S) = 0 \rightarrow \Lambda = reject$
$t_C \xleftarrow{r} \mathbb{Z}_q, T_C = g^{t_C} \mod p$
$\sigma_C := \mathsf{SIG.Sign}(sk_C, m_1||\ldots||m_8)$
$pms := T_S^{t_C} \mod p$
$ms := \mathsf{PRF}(pms, label_1||r_C||r_S)$
$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C} := \mathsf{PRF}(ms, label_2||r_C||r_S)$
$fin_C := \mathsf{PRF}(ms, label_3||m_1||\ldots||m_{10})$
$m_7 : cert_C$
$m_8 : T_C$
$m_9 : \sigma_C$
$m_{10} : flag_{enc}$
$m_{11} : (C_{11}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{mac}}^{C\to S}, \ell, H, fin_C, st_e)$
If $\mathsf{SIG.Vfy}(pk_C, \sigma_C, m_1||\ldots||m_8) = 0 \rightarrow \Lambda = reject$
$pms := T_C^{t_S} \mod p$
$ms := \mathsf{PRF}(pms, label_1||r_C||r_S)$
$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C} := \mathsf{PRF}(ms, label_2||r_C||r_S)$
$fin_S := \mathsf{PRF}(ms, label_4||m_1||\ldots||m_{12})$
$m_{12} : flag_{enc}$
$m_{13} : (C_{13}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{S\to C}, \ell, H, fin_S, st_e)$
pre-accept phase
If $fin_S \neq \mathsf{PRF}(ms, label_4||m_1||\ldots||m_{12}) \rightarrow \Lambda = reject$
If $fin_C \neq \mathsf{PRF}(ms, label_3||m_1||\ldots||m_{10}) \rightarrow \Lambda = reject$
post-accept phase
$\mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{mac}}^{C\to S}, \ell, H, data, st_e)$
$\mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{S\to C}, \ell, H, data, st_e)$

Figure 2: TLS handshake for ciphersuites `TLS_DHE_*` with client authentication

a signature computed over these values plus the two random numbers r_C and r_S. The next two messages are very simple: the `Certificate Request` message m_5 only contains a list of certificate types that the client may use to authenticate itself, and the `Server Hello Done` message m_6 does not contain any data, but consists only of a constant tag with byte-value '14' and a length value '0'.

Client Key Exchange. Having received these messages and after successful verification of σ_S the client is able to complete the key agreement and to compute the cryptographic keys. The `Client Certificate` message m_7 contains a signing certificate $cert_C$ with the public key pk_C of the client. Message m_8 is called `Client Key Exchange`, and contains the Diffie-Hellman share T_C of the client. To authenticate the client, a signature σ_C is computed on a concatenation of all previous messages (up to m_8) and padded prefixes, thus including the two random nonces and the two Diffie-Hellman shares. This signature is contained in the `Certificate Verify` message m_9.

The client is now also able to compute the *premaster secret pms*, from which all further secret values are derived. After computing the *master secret ms*, it is stored for the lifetime of the TLS session, and *pms* is erased from memory. The master secret *ms* is subsequently used, together with the two random nonces, to derive all encryption and MAC keys as well as the key confirmation message fin_C.

Client Finished. After these computations have been completed, the values are handed over to the TLS record layer of the client, which is now able to encrypt and MAC protect any data. To signal the 'start of encryption' to the server, a single message m_{10} (`Change Cipher Spec`) with byte value '1' ($flag_{enc}$) is sent unencrypted to S. The next data to be sent is `Client Finished` fin_C; message m_{11} consists of an encryption of fin_C concatenated with its MAC, where the padding described above is applied to fin_C before encryption.

Remark 2. Please note that this padding allows for (partially) known plaintext attacks on m_{11}. Thus if we analyze TLS in the BR model, the answer to a `Test` query could be tested by an adversary by simply decrypting m_{11}, and checking if the resulting plaintext has the appropriate padding. Thus TLS is not provably secure in the BR model.

Server Finished. After the server has received messages m_7, m_8, m_9, and after having checked the signature in m_9, the server can also compute *pms*, *ms*, encryption and MAC keys, and the `Server Finished` message fin_S. He can then decrypt m_{11} and check fin_C by computing the pseudo-random value on the messages sent and received by the server. If this check fails, he 'rejects' and aborts the handshake. If the check is successful, he 'accepts' and sends messages m_{12} and m_{13} to the client. If the check of fin_S on the client side is successful, the client also 'accepts'.

Encrypted Payload Transmission. The obtained keys can now be used to transmit payload data in the TLS Record Layer using a stateful symmetric encryption scheme $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$ (cf. Section 2.4). The CBC-based TLS record layer protocols work as follows. The state st_e of the encryption algorithm consists of a sequence number, which is incremented on each encryption operation. The encryption algorithm takes a message m and computes a MAC over m, the sequence counter, and some additional header data H (such as version numbers, for instance). Then message and MAC are encoded into a bit string by using a padding to a specified length l and encrypted ("MAC-then-Encode-then-Encrypt").

The state st_d of the decryption algorithm consists of a sequence number, which incremented on each decryption operation. Given a ciphertext, the algorithm decrypts and verifies the MAC using its own sequence counter. See [50] for details.

Abbreviated handshake. In our analysis, we do not consider abbreviated TLS handshakes. We note however that the server can always enforce a full handshake, and that it may also be possible to prove the security of abbreviated handshakes in our framework. Also note that abbreviated handshakes (or renegotiations) may lead to attacks on TLS, e.g. the attack in 2009 on TLS renegotation which gave an adversary access to a limited encryption oracle. We do not cover such attacks in our model as of now, see also the discussion in App. A.

Error messages. Error messages and other side channel information can be extremely useful to mount attacks on TLS. However they are outside the scope of this paper, and left as future work.

4 AKE Protocols

AKE Protocols. Let $\mathcal{K} = \{0,1\}^{\kappa}$. An authenticated key-exchange (AKE) protocol is a protocol executed among two parties. Each party maintains (at least) two internal state variables $\Lambda \in \{\texttt{accept}, \texttt{reject}\}$ and $k \in \mathcal{K}$. At some point during the protocol execution (typically after the last message according to the protocol specification has been sent) each party ends up either with an internal state $(\Lambda, k) = (\texttt{accept}, k)$ for some $k \in \mathcal{K}$, or with internal state $(\Lambda, k) = (\texttt{reject}, \emptyset)$ where $\emptyset$ denotes the empty string.

While the security models for, say, (public-key) encryption (e.g., IND-CPA or IND-CCA security), or digital signature schemes (e.g., EUF-CMA), are clean and simple, a more complex model is required to model the capabilities of active adversaries to define secure authenticated key-exchange. An important line of research [15, 20, 43, 25] dates back to Bellare and Rogaway [11], where an adversary is provided with an 'execution environment', which emulates the real-world capabilities of an active adversary. In this model, the adversary has full control over the communication network, which allows him to forward, alter, or drop any message sent by the participants, or insert new messages.

4.1 Execution Environment

Consider a scenario where there is a set of parties $\{P_1, \ldots, P_\ell\}$ for $\ell \in \mathbb{N}$, where each party $P_i \in \{P_1, \ldots, P_\ell\}$ is a potential protocol participant and each party has a long-term key pair (pk_i, sk_i)[4].

To model several sequential and parallel executions of the protocol, each party P_i is modeled by a collection of oracles $\pi_i^1, \ldots, \pi_i^d$ for $d \in \mathbb{N}$. Each oracle π_i^s represents one single process that executes an instance of the protocol. All oracles $\pi_i^1, \ldots, \pi_i^d$ representing party P_i have access to the same long-term key pair (pk_i, sk_i) of P_i and to all public keys $pk_1, \ldots, pk_\ell$. Moreover, each oracle π_i^s maintains a separate internal state

- a variable Π storing the 'destination address' d_j (not an identity nor a cryptographic identifier) of an intended communication partner P_j,[5]
- a variable $\Lambda \in \{\texttt{accept}, \texttt{reject}\}$,
- a counter Δ used to keep track of the current status of the protocol execution,
- a variable k storing the session key used for symmetric encryption between π_i^s and Π

[4]We assume that each party P_i is uniquely identified by its public key pk_i. In practice, there may be other identities that are bound to this public key, e.g. by using certificates. However, this is out of scope of this paper.

[5]Please note that we use a post-specified peer model here [21].

- and some additional temporary state variable st (which may, for instance, be used to store ephemeral Diffie-Hellman exponents or the transcript of all messages sent/received during the handshake).

When describing a protocol in the sequel, we will enumerate the protocol messages. The oracles keep track of the protocol execution by setting the counter state equal to the message number that the oracle expects to receive next, and update the counter on each message sent (e.g. $\Delta \leftarrow \Delta + 1$). The internal state of each oracle is initialized to $(\Pi, \Lambda, \Delta, k, st) = (\emptyset, \emptyset, 1, \emptyset, \emptyset)$. Furthermore, we will always assume (for simplicity) that $k = \emptyset$ if an oracle has not reached `accept`-state (yet), and contains the computed key if an oracle is in `accept`-state, so that we have

$$k \neq \emptyset \iff \Lambda = \texttt{accept}. \tag{1}$$

An adversary may interact with these oracles by issuing the following queries.

- $\mathsf{Send}(\pi_i^s, m)$: The adversary can use this query to send any message m of his own choice to oracle π_i^s. The oracle will respond according to the protocol specification, depending on its internal state. If $\Delta = 1$ and $m = (\top, d_j)$ consists of a special symbol $\top$ and a destination address d_j, then π_i^s will set $\Pi = d_j$ and respond with the first protocol message.

- $\mathsf{Reveal}(\pi_i^s)$: Oracle π_i^s responds to a Reveal-query with the contents of variable k. Note that we have $k \neq \emptyset$ if and only if $\Lambda = \texttt{accept}$, see (1).

- $\mathsf{Test}(\pi_i^s)$: This query may only be asked once throughout the game. Oracle π_i^s handles this query as follows: If the oracle has state $\Lambda \neq \texttt{accept}$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element $k_0 \overset{\$}{\leftarrow} \mathcal{K}$, sets $k_1 = k$ to the 'real' session key, and returns k_b.

The Send message enables the adversary to initiate and run an arbitrary number of protocol instances, sequential or in parallel, and provides full control over the communication between all parties. The Reveal query may be used to learn the session keys used in previous/concurrent protocol executions. The Test-query will be used to define security.

There exist variants of the above model [12, 15] that also allow $\mathsf{Corrupt}$-queries which reveal the long-lived key of a party P_i. To keep the model simple, we omit such an explicit $\mathsf{Corrupt}$-query and use that in the BR-model *static* corruptions are equivalent to *strong adaptive corruptions* (see [54, §15]). In our definition, static corruptions are modeled by letting the adversary choose a set of public-keys at the beginning of the security game. In response, all corresponding parties are marked as 'corrupted' and the adversary is provided with the corresponding secret keys. [6]

4.2 Security Definition

Bellare and Rogaway [11] have introduced the notion of *matching conversations* in order to define correctness and security of an AKE protocol precisely.
Let $T_{i,s}$ denote the transcript of all messages sent and received by process π_i^s. Let $T_{i,s}^{(-1)}$ be the transcript $T_{i,s}$ truncated by the last message.

Definition 5. We say that a processes π_i^s has a *matching conversation* to process π_j^t, if

[6]Note that in doing so we loose a factor of $\frac{1}{d^2\ell^2}$ in the reduction.

- π_i^s has sent the last message, and it holds that $T_{i,s}^{(-1)} = T_{j,t}^{(-1)}$, or
- π_j^t has sent the last message, and it holds that $T_{i,s} = T_{j,t}$.

We say that two processes π_i^s and π_j^t have *matching conversations* if π_i^s has a *matching conversation* to process π_j^t, and vice versa.

Remark 3. We remark that matching conversations in the above sense can also be seen as *post-specified session identifiers.* The 'asymmetry' of the definition (i.e., the fact that we have to distinguish which party has sent the last message) is necessary, due to the fact that protocol messages may be sent sequentially. For instance in the TLS handshake protocol (see Figure 2) the last message of the client is the 'client finished' message fin_C, and then it waits for the 'server finished' message fin_S before acceptance. In contrast, the server sends fin_S *after* receiving fin_C. Therefore the server has to 'accept' without knowing whether its last message was received by the client correctly. We have to take this into account in the definition of matching conversations, since it will later be used to define security of the protocol in presence of an active adversary that may simply drop the last protocol message.

Security of AKE protocols is now defined by requiring that (i) the protocol is a secure authentication protocol, thus any party π_i^s accepts only if there exists another party π_j^t such that π_i^s has a matching conversation to π_j^t, and (ii) the protocol is a secure key-exchange protocol, thus an adversary cannot distinguish the session key k from a random key.

AKE Game. We formally capture this notion as a game, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The challenger implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. At the beginning of the game, the challenger generates ℓ long-term key pairs (pk_i, sk_i) for all $i \in [\ell]$. The adversary receives the public keys $pk_1, \ldots, pk_\ell$ as input. Then it may submit a subset $corrupt \subseteq [\ell]$ and receives the secret keys sk_i for all $i \in corrupt$. All parties P_i with $i \in corrupt$ are said to be *corrupted.* Now the adversary may start issuing Send and Reveal queries, as well as one Test query. Finally, the adversary outputs a bit b' and terminates.

Definition 6. We say that an AKE protocol is *(t, ϵ)-secure*, if for all (possibly probabilistic) adversaries $\mathcal{A}$ running in time t and for all uncorrupted parties P_i and P_j in the AKE Game it holds that:

1. When $\mathcal{A}$ terminates, there exists no oracle π_i^s (except with probability ϵ), such that
   - π_i^s has internal state $\Lambda = \mathtt{accept}$ with $\Pi = d_j$, and
   - there is no oracle π_j^t such that π_i^s and π_j^t have matching conversations.
2. When $\mathcal{A}$ returns b' such that
   - $\mathcal{A}$ has issued a Test-query to oracle π_i^s, and
   - π_i^s has internal state $\Pi = d_j$, and
   - $\mathcal{A}$ did not issue a Reveal query to π_i^s, nor to an oracle π_j^t such that π_j^t has a matching conversation with oracle π_i^s,

   then the probability that b' equals the bit b sampled by the Test-query is bounded by

$$\left|\Pr[b = b'] - 1/2\right| \leq \epsilon.$$

Remark 4. There is a subtle difference between the model described above and the classical model of Bellare and Rogaway from [11], which we want to highlight due to its importance. Bellare and Rogaway made the restriction that the Test-query is the adversary's last. It was pointed out (see Remark 5 in [9]) that this does not guarantee security for certain applications. It is however folklore [9] that simply removing the restriction suffices to solve the issue. Accordingly, we allow the adversary explicitly to make further queries after the Test-query.

Remark 5. It is easily possible to extend the above Bellare-Rogaway model, such that also *perfect forward secrecy* of key exchange protocols is considered. This can be done by allowing to corrupt even the "test oracle" or its partner, but only *after* issuing the Test query. Though we do not consider it in detail, we are confident that all our proofs are valid in this extended model as well. A detailed analysis is left open for future extensions of this work.

5 Truncated TLS with Ephemeral Diffie-Hellman is a Secure AKE Protocol

In this section we prove the security of a modified version of the TLS handshake protocol. As discussed in the introduction, it is impossible to prove the full TLS handshake protocol secure in the BR model, since the encryption and MAC of the `Finished` messages provide a 'check value', which can be exploited by an adversary to answer the Test query correctly with probability 1.

Therefore we consider the 'truncated' TLS version from [47, 48]. In this truncated version, we assume that the `Finished` messages are sent in clear, that is, neither encrypted nor authenticated by a MAC. More precisely, we modify the TLS protocol depicted in Figure 2 such that message m_{11} contains only fin_C (instead of $\mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{mac}}^{C\to S}, \ell, H, fin_C, st_e)$), and m_{13} contains only fin_S (instead of $\mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{S\to C}, \ell, H, fin_S, st_e)$). Interestingly, this simple modification allows us to prove security in the standard Bellare-Rogaway model.

We consider an 'exact security' setting, where the success probability of all adversaries running in some time t is bounded using the concrete parameter size specified in the TLS standard.

Theorem 1. *Assume that the pseudo-random function* PRF *is* $(t, \epsilon_{\mathsf{PRF}})$-secure*, the signature scheme is* $(t, \epsilon_{\mathsf{SIG}})$-secure *and the* DDH*-problem is* $(t, \epsilon_{\mathsf{DDH}})$-hard *in* $\mathbb{Z}_q$ *with respect to the definitions in Section 2. Then the truncated ephemeral Diffie-Hellman TLS handshake protocol is a* (t', ϵ)-secure AKE *protocol with* $t \approx t'$ *and*

$$\epsilon \leq \frac{d^2\ell^2}{2^{\lambda_1 - 1}} + 2 \cdot (\ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \cdot \epsilon_{\mathsf{DDH}}) + \frac{d\ell}{2^{\mu}}$$

in the sense of Definition 6.

Proof Overview. We prove Theorem 1 in two stages. First, we show that TLS is a secure authentication protocol, that is, TLS meets Property 1.) of Definition 6. On a high level, the proof proceeds as follows.

In a first step, we ensure that no adversary can modify the Diffie-Hellman shares sent by two (uncorrupted) oracles. To this end, both oracles verify exchanged signatures and 'reject' on failure. Since *both* random nonces and the Diffie-Hellman shares are input to the digital signatures, any adversarial modification of the Diffie-Hellman shares will result in a 'reject' of one of the two oracles. The fact that these values cannot be altered by an adversary enables us in a second step to use the DDH assumption to replace the premaster secret (the seed to the PRF, which consists of the Diffie-Hellman key) with a random value. The remaining steps of the proof exploit that the PRF seed is an independent random value, since this implies that we can replace both the master secret and the returned keys with random values, assuming that the PRF is secure. Finally, observe that the PRF is used to compute the `Finished` messages. We now exploit that a PRF trivially gives rise to a MAC. As both oracles verify this MAC, we conclude that no adversary can alter any message without detection, otherwise we are able to break the security of the PRF.

Second, we show that TLS is a secure key-exchange protocol (Property 2.) of Definition 6). The proof is very similar to the above, we merely add two steps. We first replace the keys output by PRF with independent random values. Then we reduce to the sLHAE security of the record layer protocol, which is possible due to the fact that the keys now are independently random.

Proof. We prove Theorem 1 by two lemmas. Lemma 1 states that the AKE protocol meets Property 1.) of Definition 6 (authentication), Lemma 2 states that it meets Property 2.) of Definition 6 (indistinguishable keys).

5.1 Authentication

Lemma 1. *The TLS handshake protocol meets property 1.) of Definition 6.*

Proof. The proof proceeds in a *sequence of games*, following [13, 55]. The first game is the real security experiment. We then describe several intermediate games that modify the original game step-by-step, and argue that our complexity assumptions imply that each game is computationally indistinguishable from the previous one. We end up in the final game, where no adversary can break the security of the protocol. All modifications are made only to *uncorrupted* oracles whose intended partner is not corrupted, i.e., oracles whose internal state Π points to a non-corrupted party, as otherwise the adversary can trivially detect these modifications.

Let $\lambda_1, \mu \in \mathbb{N}$. Assume PRF has output bit-length μ, the exchanged random nonces have size λ_1. Let $\mathsf{break}_\tau^{(1)}$ be the event that (i) there exists oracle π_i^s reaches internal state $\Lambda = \mathtt{accept}$, but (ii) there is no oracle π_j^t such that π_i^s and π_j^t have matching conversations, in Game τ.

Game 0. This game equals the AKE security experiment described in Section 4.2.

Game 1. In this game, the challenger proceeds exactly like the challenger in Game 0, except that we add an abort rule. The challenger raises event $\mathsf{abort}_{\mathsf{nonce}}$ and aborts, if during the simulation a *pair* of nonces (r_C, r_S) appears twice.

Since both games proceed identical until $\mathsf{abort}_{\mathsf{nonce}}$ is raised, we have

$$\left|\Pr[\mathsf{break}_0^{(1)}] - \Pr[\mathsf{break}_1^{(1)}]\right| \le \Pr[\mathsf{abort}_{\mathsf{nonce}}].$$

All oracles sample r_C or r_S uniformly random from $\{0,1\}^{\lambda_1}$. Thus, by applying the birthday bound and the fact that the adversary has access to at most $d\ell$ oracles, we have

$$\Pr[\mathsf{abort}_{\mathsf{nonce}}] \le \frac{d^2\ell^2}{2^{\lambda_1}}.$$

Note, that each oracle sends and receives exactly one nonce and that the combination of both nonces is unique in Game 1 (as the game is aborted otherwise). Therefore we may regard a tuple (r_C, r_S) as a 'session identifier' for one particular oracle. We will use in the sequel that this 'session identifier' is used as input to both the signing algorithm (providing authentication of the communication partner) and to the computation of the `Finished` message (which ensure matching conversations).

Game 2. In this game we want to make sure that each accepting oracle receives as input *exactly* the nonce r_C or r_S and the 'Diffie-Hellman key exchange value' T_C or T_S which was chosen and sent by its 'partner' oracle. Here we can use the fact that the random nonces r_C and r_S are unique due to Game 1 and that the tuple (r_C, r_S) is included together with (at least one of) the Diffie-Hellman shares in both signatures issued by the partners.

Technically, we add another abort condition. The challenger proceeds exactly as before, but raises event $\mathsf{abort_{sig}}$ and aborts if there exists an oracle π_i^s such that π_i^s 'accepts', and

- π_i^s has session identifier (r_C, r_S) and Diffie-Hellman shares (T_C, T_S)
- there is no oracle π_j^t which has the same combination of session identifier (r_C, r_S) and Diffie-Hellman shares (T_C, T_S)
- but the signature received by π_i^s and computed over T_S or (T_C, T_S) verifies correctly under the long-term public key pk_j of π_j^t.[7],

Clearly we have

$$\left|\Pr[\mathsf{break}_1^{(1)}] - \Pr[\mathsf{break}_2^{(1)}]\right| \leq \Pr[\mathsf{abort_{sig}}].$$

To show that $\Pr[\mathsf{abort_{sig}}]$ is negligible, we construct a signature forger as follows. The forger receives as input a public key pk^* and simulates the challenger for $\mathcal{A}$. It guesses an index $\phi \xleftarrow{\$} [\ell]$, sets $pk_\phi = pk^*$, and generates all long-term public/secret keys as before. Then it proceeds as the challenger in Game 2, except that it uses its chosen-message oracle to generate a signature under pk_ϕ when necessary.

When $\mathsf{abort_{sig}}$ is raised, then this means that

- π_i^s has received as input a signature containing (r_C, r_S) and either T_S or (T_C, T_S)
- no oracle π_j^t has ever issued a signature containing (r_C, r_S) and either T_S or (T_C, T_S)
- but the signature verifies correctly under pk_j.

If $\phi = j$, which happens with non-negligible probability $1/\ell$, then the forger can use the signature received by π_i^s to break the EUF-CMA security of the signature scheme with success probability ϵ_{SIG}, so $\frac{\Pr[\mathsf{abort_{sig}}]}{\ell} \leq \epsilon_{\mathsf{SIG}}$. Therefore if $\Pr[\mathsf{abort_{sig}}]$ is not negligible, then ϵ_{SIG} is not negligible as well and we have

$$\left|\Pr[\mathsf{break}_1^{(1)}] - \Pr[\mathsf{break}_2^{(1)}]\right| \leq \ell \cdot \epsilon_{\mathsf{PRF}}.$$

Note that in Game 2 each accepting oracle π_i^s has a *unique* 'partner' oracle π_j^t sharing the same session identifier (r_C, r_S), as otherwise the game is aborted. Note also that in Game 2 *each* Diffie-Hellman share received by an accepting oracle π_i^s is exactly the value that was sent by the partner

[7]It is important to note that the signature sent by a 'server' oracle is only computed over the Diffie-Hellman share of this oracle T_S, whereas the signature sent by a 'client' oracle protects both shares (T_C, T_S).

oracle π_j^t of π_i^s. Both oracles are implemented by the challenger, thus, in Game 2 the challenger has full control over all Diffie-Hellman keys computed by all accepting oracles throughout the game. This enables us to exchange the premaster secret values in Game 3.

Game 3. In this game, we replace the Diffie-Hellman keys computed by all oracles with uniformly random group elements of the subgroup. The fact that the challenger has full control over the established Diffie-Hellman keys, due to the modifications introduced in the previous games, provides us with the leverage to prove indistinguishability under the Decisional Diffie-Hellman assumption.

Technically, the challenger in Game 3 proceeds as before, but whenever a premaster secret pms has to be computed as $pms = g^{t_C t_S}$, the challenger replaces this value with a uniformly random value $\widetilde{pms} = g^r$ for $r \xleftarrow{\$} \mathbb{Z}_q$, which is in the following used by both partner oracles.

Suppose there exists an algorithm distinguishing Game 3 from Game 2. Then we can construct an algorithm $\mathcal{B}$ solving the DDH problem as follows. Algorithm $\mathcal{B}$ receives as input a DDH challenge (g, g^a, g^b, g^c). It runs the (efficient) algorithm described in [4, Lemma 5.2] to generate $3d\ell$ group elements[8]

$$(g^{a_1}, g^{b_1}, g^{c_1}), \ldots, (g^{a_{d\ell}}, g^{b_{d\ell}}, g^{c_{d\ell}})$$

such that

- $c_i = a_i b_i$ for all $i \in d\ell$, if $c = ab$, and
- the vector $(g^{a_i}, g^{b_i}, g^{c_i})_{i \,\in\, d\ell}$ is uniformly distributed over $(\mathbb{Z}_p^*)^{3d\ell}$, if $c \neq ab$.

The challenger uses the values g^{a_i} instead of the g^{t_S} values chosen by a 'server' oracle, and the corresponding value g^{b_i} instead of the g^{t_C} for the 'client' oracle sharing the same session identifier (r_C, r_S) with the 'server' oracle. Instead of computing the Diffie-Hellman key as in Game 2, it sets $pms = g^{c_i}$ both for the 'client' and the 'server' oracle. Now if $c_i = a_i b_i$, then this game proceeds exactly like Game 2, while if c_i is random than this game proceeds exactly like Game 3. Now if the algorithm that is able to distinguish Game 3 from Game 2 outputs 1 (meaning this game proceeds like Game 3), $\mathcal{B}$ outputs 1 as answer to the DDH challenge (meaning c_i is chosen at random), otherwise $\mathcal{B}$ outputs 0. The DDH assumption therefore implies that

$$\left|\Pr[\mathsf{break}_2^{(1)}] - \Pr[\mathsf{break}_3^{(1)}]\right| \leq \epsilon_{\mathsf{DDH}}$$

Note that in Game 3 all premaster secrets of accepting oracles are uniformly random, and independent of any messages sent throughout the game. This will provide us with the leverage to replace the function $\mathsf{PRF}(\widetilde{pms}, \cdot)$ with a truly random function, whose output is also independent of any messages sent, in the next game.

Game 4. In Game 4 we make use of the fact that the premaster secret $\widetilde{pms}$ is chosen uniformly random, and independent of any message sent. We thus replace the value $ms = \mathsf{PRF}(\widetilde{pms}, \cdot)$ with a random value sampled from a truly random function $\widetilde{ms} = F_{\widetilde{pms}}$, since PRF is assumed to be a pseudo-random function which is secure according to Definition 3.

We prove this by a hybrid argument and define a sequence of hybrid games $H_0, \ldots, H_n$, such that hybrid H_0 equals Game 3 and H_n equals Game 4. Then we argue that hybrid H_{i-1} is

[8]Recall that $d\ell$ equals the number of oracles the adversary may query.

indistinguishable from H_i for $i \in \{1, \ldots, n\}$, except for some small probability ϵ_{PRF}. We define the hybrids as follows: H_0 behaves exactly as Game 3. Then in hybrid i we replace the computed master secret ms with a uniformly random value $\widetilde{ms}$ for oracle π_j^t, where j and t are uniquely identified by index i. Hybrid H_n then behaves exactly as Game 4. Let E_i denote the event that $\mathcal{A}$ outputs 1 in hybrid i. Suppose for contradiction

$$|\Pr[E_0] - \Pr[E_n]| > n \cdot \epsilon_{\mathsf{PRF}},$$

that is the difference in the success probability of $\mathcal{A}$ in hybrid H_0 compared to the success probability in H_n is negligible except for probability $n \cdot \epsilon_{\mathsf{PRF}}$.[9] In this case there must exist an index i such that $|\Pr[E_{i-1}] - \Pr[E_i]| > \epsilon_{\mathsf{PRF}}$.

Suppose there exists an algorithm $\mathcal{D}$ able to distinguish between two hybrids H_{i-1} and H_i (with probability $|\Pr[E_{i-1}] - \Pr[E_i]| = \epsilon_{\mathsf{PRF}}$). Then we can construct an adversary $\mathcal{B}$ breaking the security of the PRF as follows. At first, $\mathcal{B}$, guesses an index $i \in [n]$ (and with probability $\frac{1}{n}$ this index corresponds to the index i where $|\Pr[E_{i-1}] - \Pr[E_i]| = max_i\, |\Pr[E_{i-1}] - \Pr[E_i]|$) and inputs (r_C, r_S) to the PRF-challenger. The challenger computes $z_0 = \mathsf{PRF}(\widetilde{pms}, label_1 || r_C || r_S)$ and $z_1 = F_{\widetilde{pms}}$, tosses a coin $b = 0/1$ and outputs z_b to the adversary. Then $\mathcal{B}$ sets $ms = z_b$ for party π_j^t indexed by i. Now, if $\mathcal{D}$ outputs 1, $\mathcal{B}$ returns 1 to the challenger, otherwise it returns 0. If $\mathcal{B}$ guessed the correct index, then the answer of $\mathcal{B}$ is correct with probability $max_i\, |\Pr[E_{i-1}] - \Pr[E_i]|$

Thus we have

$$\left|\Pr[\mathsf{break}_3^{(1)}] - \Pr[\mathsf{break}_4^{(1)}]\right| \le d\ell \cdot \epsilon_{\mathsf{PRF}}$$

Game 5. In this game we make a modification similar to the changes introduced in Game 4. This time we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ with a random function, which is possible since $\widetilde{ms}$ is independent and uniformly random.

Instead of evaluating the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ to compute and verify the `Finished` messages, the challenger in Game 5 samples a truly random function $F_{\widetilde{ms}}$ with output bit-length μ. Of course the same random function is used for both partner oracles sharing the same $\widetilde{ms}$. In particular, this function is used to compute the `Finished` messages by both partner oracles.

With the same arguments as in Game 4 distinguishing Game 5 from Game 4 implies an algorithm breaking the security of the pseudo-random function PRF, thus

$$\left|\Pr[\mathsf{break}_4^{(1)}] - \Pr[\mathsf{break}_5^{(1)}]\right| \le d\ell \cdot \epsilon_{\mathsf{PRF}}$$

Game 6. Finally we use that the full transcript of all messages sent and received is used to compute the `Finished` messages, and that `Finished` messages are computed by evaluating a truly random function due to Game 5, to show that any adversary has only a negligible probability of making an oracle accept without a partner oracle having a matching conversation.

Thus, this game proceeds exactly like the previous game, except that the challenger now raises event $\mathsf{abort_{fin}}$ and aborts if an oracle π_i^s accepts, but there is no oracle π_j^t having a matching conversation to π_s.

The `Finished` messages are computed by evaluating a truly random function $F_{\widetilde{ms}}$, which is only accessible to the partner oracles sharing $\widetilde{ms}$, and the full transcript containing all previous

[9]Note that $n \le d\ell$.

messages is used to compute the `Finished` messages. If there is no oracle having a matching conversation to π_i^s, the adversary receives no information about $F_{\widetilde{ms}}(preamble||m_1||\cdots||m_{10})$ (resp. $F_{\widetilde{ms}}(preamble||m_1||\cdots||m_{12})$). Therefore we have $\Pr[\mathsf{break}_6^{(1)}] = \frac{1}{2^\mu}$ and

$$\left|\Pr[\mathsf{break}_5^{(1)}] - \Pr[\mathsf{break}_6^{(1)}]\right| \le \frac{d\ell}{2^\mu}.$$

Collecting probabilities from Game 0 to Game 6 proves Lemma 1.

Together we have

$$\left|\Pr[\mathsf{break}_0^{(1)}] - \Pr[\mathsf{break}_6^{(1)}]\right| \le \frac{d^2\ell^2}{2^{\lambda_1}} + \ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{DDH}} + \frac{d\ell}{2^\mu}.$$

□

5.2 Indistinguishability of Keys

Lemma 2. *The TLS handshake protocol meets property 2.) of Definition 6.*

PROOF. Let $\mathsf{break}_\tau^{(2)}$ denote the event that the $\mathcal{A}$ returns a bit b' such that $\mathcal{A}$ has issued a Test-query to oracle π_i^s, and

- $\mathcal{A}$ did not issue a Reveal query to π_i^s, nor to an oracle π_j^t having a matching conversation with oracle π_i^s,
- party P_i is not corrupted and for any oracle π_j^t having a matching conversation to π_i^s, it holds that Party P_j is not corrupted as well,

and b' equals the bit b sampled by the Test-query, in Game τ.

Game 0. This game equals the AKE security experiment described in Section 4.2.

Game 1. This game equals Game 4 from the proof of Lemma 1. With the same arguments as in the proof of Lemma 1, we have

$$\left|\Pr[\mathsf{break}_0^{(2)}] - \Pr[\mathsf{break}_1^{(2)}]\right| \le \frac{d^2\ell^2}{2^{\lambda_1}} + \ell \cdot \epsilon_{\mathsf{SIG}} + d\ell \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{DDH}}.$$

Game 2. Finally, we replace the key $K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C} = \mathsf{PRF}(\widetilde{ms}, label_2||r_C||r_S)$ with a uniformly random key vector. The fact that the seed to the PRF $\widetilde{ms}$ is uniformly random and independent of any messages sent, enables us to argue that Game 2 is indistinguishable from Game 1 assuming that the PRF is secure.

Technically, this game proceeds exactly like the previous game, except that the challenger now chooses the vector

$$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C}$$

uniformly random. Again an algorithm distinguishing Game 2 from Game 1 implies an algorithm breaking the security of the pseudo-random function PRF. Thus we have

$$\left|\Pr[\mathsf{break}_1^{(2)}] - \Pr[\mathsf{break}_2^{(2)}]\right| \leq d\ell \cdot \epsilon_{\mathsf{PRF}}.$$

Note that in Game 2 the response to the Test query consists always of an uniformly random key, independent of the bit b sampled in the Test query. Thus we have

$$\Pr[\mathsf{break}_2^{(2)}] = 1/2.$$

Collecting probabilities from Game 0 to Game 2 proves Lemma 2.

Together we have

$$\left|\Pr[\mathsf{break}_0^{(2)}] - \Pr[\mathsf{break}_2^{(2)}]\right| \leq \frac{d^2\ell^2}{2^{\lambda_1}} + \ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{DDH}}.$$

□

6 ACCE Protocols

An *authenticated and confidential channel establishment* (ACCE) protocol is a protocol executed among two parties. The protocol consists of two phases, called the 'pre-accept' stage and the 'post-accept' stage.

Pre-accept stage. In this stage a 'handshake protocol' is executed. In terms of functionality this protocol is an AKE protocol as in Section 4.2, that is, both communication partners are mutually authenticated, and a session key k is established. However, it need not necessarily meet the security definition for AKE protocols (Definition 6). This stage ends, when both communication partners reach an `accept` state.

Post-accept stage. This stage is entered, when both communication partners reach `accept` state. In this stage data can be transmitted, encrypted with a stateful symmetric encryption scheme $\mathsf{SE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$ under the key k established in the pre-accept phase.

The prime example for an ACCE protocol is TLS. Here, the pre-accept stage consists of the TLS handshake protocol. Then both parties enter the post-accept stage, where data is transmitted using the MAC-then-encrypt paradigm.

Defining ACCE. To define security of ACCE protocols, we combine the Bellare-Rogaway model for authenticated key exchange (see Section 4) with length-hiding stateful encryption in the sense of [50]. Technically, we provide a slightly modified execution environment that extends the types of queries an adversary may issue.

6.1 Execution environment

The execution environment is very similar to the model for AKE from Section 4, except for a simple modification. We extend the model such that in the post-accept stage an adversary is also able to 'inject' arbitrary chosen-plaintexts by making an $\mathsf{Encrypt}$ query,[10] and chosen-ciphertexts by making a $\mathsf{Decrypt}$ query. Moreover, each oracle π_i^s samples a random bit $b_i^s \xleftarrow{\$} \{0,1\}$ at the beginning of the game.

Execution model. An adversary may interact with the provided oracles by issuing the following queries.

- $\mathsf{Send}^{\mathsf{pre}}(\pi_i^s, m)$: This query is identical to the Send-query in the BR model described in Section 4, except that it replies with an error symbol $\perp$ if oracle π_i^s has state $\Lambda = \mathtt{accept}$. ($\mathsf{Send}$-queries in accept-state are handled by the $\mathsf{Decrypt}$ query below).
- $\mathsf{Reveal}(\pi_i^s)$: This query is identical to the Reveal-query in the BR model.
- $\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \ell, H)$: This query depends on the random bit $b_i^s \xleftarrow{\$} \{0,1\}$ sampled by π_i^s at the beginning of the game. It takes as input two messages m_0 and m_1 and header data H. It maintains a counter i which is initialized to 0, and proceeds as depicted in Figure 3.
- $\mathsf{Decrypt}(\pi_j^t, C, H)$: This query takes as input a ciphertext C and header data H. If $\Lambda \neq \mathtt{accept}$ then π_j^t returns $\perp$. Otherwise, it proceeds as depicted in Figure 3.

Figure 3: $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ oracles in the ACCE security experiment.

```
Encrypt($\pi_i^s, m_0, m_1, \ell, H$):
$i := i + 1$
$(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k_i^s, \ell, H, m_0, st_e)$
$(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k_i^s, \ell, H, m_1, st_e)$
If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return $\perp$
$(C_i, st_e) := (C^{(b_i^s)}, st_e^{(b_i^s)})$
Return $C_i$
```

```
Decrypt($\pi_j^t, C, H$):
$j := j + 1$
If $b = 0$, then return $\perp$
$(m, st_d) = \mathsf{StE.Dec}(k_j^t, H, C, st_d)$
If $j > i$ or $C \neq C_j$, then phase := 1
If phase = 1 then return $m$
```

Here we denote with k_a^b the value stored in the internal variable k of oracle π_a^b.

6.2 Security Definition

Security of ACCE protocols is defined by requiring that (i) the protocol is a secure authentication protocol, thus any party π_i^s reaches the post-accept state only if there exists another party π_j^t such that π_i^s has a matching conversation (in the sense of Definition 5) to π_j^t, and (ii) data transmitted in the post-accept over a secure channel in the sense of Definition 4.

Again this notion is captured by a game, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The challenger implements the collection oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. At the beginning of the game, the challenger generates ℓ long-term key pairs (pk_i, sk_i) for all $i \in [\ell]$. The adversary receives the public keys $pk_1, \ldots, pk_\ell$ as input. Then it may submit a subset *corrupt* $\subseteq [\ell]$ and receives the

[10]This models that an adversary may trick one party into sending some adversarially chosen data. A practical example for this attack scenario are cross-site request forgeries [61] on web servers, or Bard's chosen-plaintext attacks on SSL3.0 [2, 3].

secret keys sk_i for all $i \in corrupt$. All parties P_i with $i \in corrupt$ are said to be *corrupted.* Now the adversary may start issuing Send, Reveal, Encrypt and Decrypt queries. Finally, the adversary outputs a triple (i, s, b') and terminates.

Definition 7. We say that an ACCE protocol is *secure*, if for all (possibly probabilistic) adversaries $\mathcal{A}$ and for all uncorrupted parties P_i and P_j holds that:

1. When $\mathcal{A}$ terminates, there exists no oracle π_i^s (except with probability ϵ), such that:
   - π_i^s has internal state $\Lambda = \texttt{accept}$ with $\Pi = d_j$,
   - there is no oracle π_j^t such that π_i^s and π_j^t have matching conversations.
2. When $\mathcal{A}$ terminates and outputs a triple (i, s, b') such that
   - π_i^s has internal state $\Pi = d_j$,
   - $\mathcal{A}$ did not issue a Reveal query to π_i^s, nor to an oracle π_j^t, such that π_i^s and π_j^t have matching conversations,

   then the probability that b' equals b_i^s is bounded by

$$\left|\Pr[b_i^s = b'] - 1/2\right| \leq \epsilon.$$

6.3 Relation to the classical AKE Security Definition from Section 4.2

Note that an ACCE protocol can be constructed in a two-step approach.

1. (AKE part) First an authenticated key-exchange (AKE) protocol is executed. This protocol guarantees the authenticity of the communication partner, and provides a cryptographically 'good' (i.e., indistinguishable from random for the adversary) session key.
2. (Symmetric part) The session key is then used in a symmetric encryption scheme providing integrity and confidentiality.

This modular approach is simple and generic, and therefore appealing. It can be shown formally that this two-step approach yields a secure ACCE protocol, if the 'AKE part' meets the Bellare-Rogaway security definition from Section 4.2, and the 'symmetric part' consists of a suitable authenticated symmetric encryption scheme (e.g. secure according to Definition 4).

However, if the purpose of the protocol is the establishment of an authenticated confidential channel, then it is not necessary that the 'AKE-part' of the protocol provides full indistinguishability of *session keys.* It actually would suffice if *encrypted messages* are indistinguishable, and *cannot be altered* by an adversary. These requirements are strictly weaker than indistinguishability of keys in the sense of Bellare-Rogaway, and thus are easier to achieve (possibly from weaker hardness assumptions, or by more efficient protocols). Moreover, the BR-model does not capture many practical protocols, with TLS being the most prominent example.

7 TLS with Ephemeral Diffie-Hellman is a Secure ACCE Protocol

Theorem 2. *Assume that the pseudo-random function* PRF *is* $(t, \epsilon_{\mathsf{PRF}})$-secure*, the signature scheme is* $(t, \epsilon_{\mathsf{SIG}})$-secure *and the* DDH*-problem is* $(t, \epsilon_{\mathsf{DDH}})$-hard *in* $\mathbb{Z}_q$ *with respect to the definitions in Section 2. Suppose that the stateful symmetric encryption scheme is* $(t, \epsilon_{\mathsf{sLHAE}})$*-secure. Then TLS with ephemeral Diffie-Hellman is a* (t', ϵ)-secure AKE *protocol with* $t \approx t'$ *and* $\epsilon \leq \frac{d^2\ell^2}{2^{\lambda_1 - 1}} + 2 \cdot (\ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \cdot \epsilon_{\mathsf{DDH}}) + \frac{d\ell}{2^\mu} + d\ell \cdot \epsilon_{\mathsf{sLHAE}}$ *in the sense of Definition 7.*

We prove Theorem 2 via the following lemmas.

Lemma 3. *The TLS protocol meets property 1.) of Definition 7.*

The proof of this lemma is almost identical to the proof of Lemma 1, and therefore omitted.

Lemma 4. *The TLS protocol meets property 2.) of Definition 7.*

Proof. Let $\mathsf{break}_\tau^{(3)}$ be the event that the attacker returns a triple (i, s, b') such that

- $\mathcal{A}$ has not made an $\mathsf{Encrypt}$ query to π_i^s that returned C^*, nor to an oracle π_j^t having a matching conversation with π_i^s,
- $\mathcal{A}$ has not made a Reveal query to π_i^s, nor to an oracle π_j^t having a matching conversation with π_i^s,
- Party P_i is not corrupted and for any oracle π_j^t having a matching conversation to π_i^s, it holds that party P_j is not corrupted as well,

and b' equals the bit b_i^s sampled by oracle π_i^s in Game τ.

Game 0. This game equals the ACCE security experiment described in Section 4.2.

Game 1. This game equals Game 2 in the proof of Lemma 2. With the same arguments as in the proof of Lemma 2, we have that no adversary can distinguish between Game 1 and Game 0. Note that in particular the keys K_{mac} and K_{enc} are now chosen uniformly at random.

$$\left|\Pr[\mathsf{break}_0^{(3)}] - \Pr[\mathsf{break}_1^{(3)}]\right| \leq \frac{d^2\ell^2}{2^{\lambda_1}} + \ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{DDH}}.$$

Game 2. Observe that the key $K = K_{\mathsf{enc}} || K_{\mathsf{mac}}$ used in the stateful symmetric encryption scheme is now chosen uniformly at random for all oracles π_i^s with $\Pi = d_j$ such that P_i and P_j are not corrupted.

In this game we construct a simulator $\mathcal{B}$ that uses a successful ACCE attacker $\mathcal{A}$ to break the security of the underlying sLHAE secure symmetric encryption scheme (Definition 4). By assumption, the simulator $\mathcal{B}$ is given access to an encryption oracle $\mathsf{Encrypt}$ and a decryption oracle $\mathsf{Decrypt}$. $\mathcal{B}$ uniformly random draws $i' \in [l] \backslash corrupt$ and $s' \in [d]$ and embeds the sLHAE experiment for oracle $\pi_{s'}^{i'}$ by simply forwarding all $\mathsf{Encrypt}(\pi_{i'}^{s'}, \cdot)$ queries to $\mathsf{Encrypt}$, and all $\mathsf{Decrypt}(\pi_j^t, \cdot)$ queries to $\mathsf{Decrypt}$, where π_j^t is an oracle having a matching conversation in pre-accept stage with $\pi_{i'}^{s'}$. Otherwise it proceeds as the challenger in Game 1.

Observe that the values generated in this game are exactly distributed as in the previous game. We have

$$\Pr[\mathsf{break}_1^{(3)}] = \Pr[\mathsf{break}_2^{(3)}].$$

Finally $\mathcal{A}$ outputs a triple (i, s, b'). If $i = i'$ and $s = s'$, then $\mathcal{B}$ forwards b' to the sLHAE challenger. Otherwise it outputs a random bit. Since the simulator essentially relays all messages it is very easy to see that an attacker $\mathcal{A}$ having success probability $1/2 + \epsilon$ yields an attacker $\mathcal{B}$ against the sLHAE security of the encryption scheme with success probability at least $1/2 + \epsilon/(\ell d)$.

Since by assumption any attacker has at most negligible success probability in breaking the sLHAE security of the symmetric encryption scheme, we have

$$\Pr[\mathsf{break}_2^{(3)}] \leq 1/2 + d\ell \cdot \epsilon_{\mathsf{sLHAE}}.$$

Collecting probabilities we get that

$$\left|\Pr[\mathsf{break}_0^{(3)}] - \Pr[\mathsf{break}_2^{(3)}]\right| \leq 1/2 + \frac{d^2\ell^2}{2^{\lambda_1}} + \ell \cdot \epsilon_{\mathsf{SIG}} + 2 \cdot d\ell \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{DDH}} + d\ell \cdot \epsilon_{\mathsf{sLHAE}}.$$

□

8 Conclusion

In this paper we have shown that TLS with ephemeral Diffie-Hellman (DHE) provides a sound framework for the secure establishment of confidential and authenticated channels. Our result is somewhat surprising: contrary to what previous analyses might suggest the random oracle model is not required to show that the composition of cryptographic building blocks in TLS is secure. In particular, our work shows that TLS can easily be made provably secure without random oracles (under standard security assumptions) when simply relying on ciphersuites based on DHE and CBC-mode, and signature schemes that are provably secure in the standard model. Moreover our result sheds a critical light on the latest revision phase where the mandatory authentication method was changed from Diffie-Hellman key exchange to encrypted key transport, at least from a provable-security point-of-view.

The design of the DHE handshake seems to support our proof idea of reducing active adversaries to passive attackers very naturally. This is not due to the key transport mechanism itself but rather to the fact that DHE uses signatures computed over all previously exchanged messages to authenticate the protocol parties. Our proof essentially exploits that this authentication mechanism at the same time also protects the first phase of the handshake from adversarial modifications. We cannot identify a similarly straight-forward approach for encrypted key transport, as server authentication is done rather implicitly when verifying the `Finished` messages. Likewise, the static Diffie-Hellman key exchange does not use signatures over the first handshake messages. Thus our proof technique does not apply.

We have analyzed one particular family of TLS ciphersuites, namely those based on ephemeral Diffie-Hellman. It would be possible in practice to configure servers such that only these ciphersuites are used (recall that `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` is mandatory for TLS 1.0 and 1.1). However, it would also be nice to be able to analyze e.g. the RSA-based ciphersuites, in particular since the current version 1.2 specifies such a ciphersuite as mandatory. Any standard-model approach we can think of would require IND-CCA security of the cipher used to transport the premaster secret from the client to the server, as otherwise we cannot simulate protocol executions while still being able to argue with indistinguishability of premaster secrets. But unfortunately it is well-known that the RSA-PKCS v1.5 cipher used in TLS is vulnerable to chosen-ciphertext attacks [16]. Other papers on TLS assume RSA-PKCS v2.0 (RSA-OAEP instead), which is however not used in TLS, and is also only provably secure with random oracles.

The whole TLS protocol suite is very flexible, for instance it allows to negotiate ciphersuites at the beginning of the handshake, or to resume sessions using an abbreviated handshake. We need to leave an analysis of these features for future work, since the complexity of the protocol and security model grows dramatically.

The goal of this work is to analyze TLS-DHE on the protocol layer. As common in cryptographic protocol analyses, we therefore have ignored implementational issues like error messages, which of course might also be used to break the security of the protocol. We leave it as an interesting open question to find an adequate approach for modeling such side-channels in complex scenarios like AKE involving many parties and parallel, sequential, and concurrent executions.

So clearly the security analysis of TLS is not finished yet, there are still many open questions. However, we find it interesting that it is possible to get some evidence towards the security of some TLS ciphersuites without random oracles and other idealizations. We consider this as a strong indicator for the soundness of the TLS protocol framework. We believe that future revisions of the TLS standard should be guided to the establishment of an implementation that is provably secure – ideally in the standard model. This work offers a solid start in this direction.

References

[1] Boaz Barak, Yehuda Lindell, and Tal Rabin. Protocol initialization for the framework of universal composability. Cryptology ePrint Archive, Report 2004/006, 2004. `http://eprint.iacr.org/`.

[2] Gregory V. Bard. The vulnerability of ssl to chosen plaintext attack. Cryptology ePrint Archive, Report 2004/111, 2004. `http://eprint.iacr.org/`.

[3] Gregory V. Bard. A challenging but feasible blockwise-adaptive chosen-plaintext attack on ssl. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT*, pages 99–109. INSTICC Press, 2006.

[4] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, May 2000.

[5] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 1–11. ACM Press, November 2002.

[6] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Trans. Inf. Syst. Secur.*, 7:206–241, May 2004.

[7] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, December 2000.

[8] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.

[9] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.

[10] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[11] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1994.

[12] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, May / June 1995.

[13] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[14] Karthikeyan Bhargavan, Cédric Fournet, Ricardo Corin, and Eugen Zalinescu. Cryptographically verified implementations for TLS. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 459–468. ACM Press, October 2008.

[15] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, December 1997.

[16] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, August 1998.

[17] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[18] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment.* Information Security and Cryptography. Springer-Veriag, Berlin,Germany, 2003.

[19] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

[20] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, May 2001.

[21] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161. Springer, August 2002. `http://eprint.iacr.org/2002/120/`.

[22] S. Chaki and A. Datta. Aspier: An automated framework for verifying security protocol implementations. In *Computer Security Foundations Symposium, 2009. CSF '09. 22nd IEEE*, pages 172 –185, july 2009.

[23] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Examining indistinguishability-based proof models for key establishment protocols. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 585–604. Springer, December 2005.

[24] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New attacks on PKCS#1 v1.5 encryption. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 369–381. Springer, May 2000.

[25] Cas J. F. Cremers. Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09: 7th International Conference on Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages 20–33. Springer, June 2009.

[26] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746.

[27] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746.

[28] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878.

[29] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[30] D. Eastlake 3rd and T. Hansen. US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634 (Informational), July 2006.

[31] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), September 2001. Updated by RFC 4634.

[32] Marc Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer, January 2003.

[33] Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08: 3rd Conference on Computer and Communications Security*, pages 21–32. ACM Press, March 2008.

[34] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally Composable Security Analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec*, volume 5324 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2008.

[35] Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, August 2001.

[36] Jakob Jonsson and Burton S. Kaliski Jr. On the security of RSA encryption in TLS. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 127–142. Springer, August 2002.

[37] B. Kaliski. PKCS #1: RSA Encryption Version 1.5. RFC 2313 (Informational), March 1998. Obsoleted by RFC 2437.

[38] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313. Springer, August 2010.

[39] Eike Kiltz and Krzysztof Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 389–406. Springer, April 2009.

[40] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, August 2001.

[41] Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, August 2005.

[42] Ralf Kuesters and Max Tuengerthal. Composition theorems without pre-established session identifiers. ACM CCS 2011, 2011.

[43] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16, 2007.

[44] Gary Locke and Patrick Gallagher. FIPS PUB 186-3 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS), 2009.

[45] Ueli Maurer and Björn Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 505–515. ACM Press, October 2010.

[46] John C. Mitchell. Finite-state analysis of security protocols. In *CAV*, pages 71–76, 1998.

[47] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 55–73. Springer, December 2008.

[48] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The TLS handshake protocol: A modular analysis. *Journal of Cryptology*, 23(2):187–223, April 2010.

[49] Kazuhiro Ogata and Kokichi Futatsugi. Equational Approach to Formal Analysis of TLS. In *ICDCS*, pages 795–804. IEEE Computer Society, 2005.

[50] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *The 17th Annual International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2011*, LNCS. Springer Verlag, December 2011.

[51] Lawrence C. Paulson. Inductive Analysis of the Internet Protocol TLS. *ACM Trans. Inf. Syst. Secur.*, 2(3):332–351, 1999.

[52] David Pointcheval and Serge Vaudenay. On Provable Security for Digital Signature Algorithms. Technical report, Ecole Normale Superieure, 1996.

[53] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992.

[54] Victor Shoup. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999. `http://eprint.iacr.org/`.

[55] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, Nov 2004.

[56] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy*, pages 19–30, 2002.

[57] Serge Vaudenay. The Security of DSA and ECDSA. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 309–323, 2003.

[58] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 protocol. In *In Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 29–40. USENIX Association, 1996.

[59] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

[60] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *IEEE Symposium on Security and Privacy*, pages 35–49. IEEE Computer Society, 2008.

[61] William Zeller and Edward W. Felten. Cross-site request forgeries: Exploitation and prevention. Technical report, October 2008. Available at http://from.bz/public/documents/publications/csrf.pdf.

A Choosing the Right Model

Authenticated key agreement (AKE) is a basic building block in modern cryptography. Many secure protocols for two-party and group key agreement have been proposed, including generic compilers that transform simple key agreement protocols into authenticated key agreement protocols, with many additional security properties. However, since many different formal models for different purposes exist, choice of the right model is not an easy task, and must be considered carefully.

The main guideline for this choice is the fact that we cannot modify any detail of the TLS protocol, nor of the network protocols preceding it.

First, we need a model were *entity authentication* is addressed as a security goal. This goal is often omitted in newer models, in order to make them suitable for two-party authenticated key agreement protocols [41]. However, explicit authentication is an important security goal for TLS, since in many practical applications authentication is more important than encryption. For example, in a Single Sign-On scenario, an encrypted security token may be passed from the identity provider through the browser to a relying party. Since the security token itself is encrypted, confidentiality is not an issue, but moreover the authenticity of the channel through which this token was received.

Second, there is no way to modularize the security proof of TLS in the sense of [20], since several protocol messages of TLS come without authenticator. Thus we cannot use the authenticated link model (AM).

Third, we have chosen not to use a Universal Composability (UC) [19] approach. We think that a formalization in the UC model first requires a thorough analysis in the standard model, otherwise the modeling of the ideal functionalities will either be wrong, or miss important security aspects of TLS. However, since the exchange of nonces r_C and r_S in the first two messages of the TLS handshake can be regarded as an instantiation of the Barak compiler [1], it is in principle possible to model TLS within the UC framework.

On the other hand, we have to make a choice about the enhanced adversarial capabilities newer models offer. We allow for $\mathsf{RevealKey}$ queries, but do not take into account $\mathsf{RevealState}$ queries. The reason for this omittance is that in TLS there are several successive internal states: Computation of the premaster secret, computation of the master secret, computation of the session keys. After transition from one state to another, internal data is erased. So to be precise, we would have to specify several different RevealState queries, which would have rendered the paper unreadable.

Remark: Please note that our proof remains valid even if different types of RevealState queries are allowed, only the simulation of the Diffie-Hellman game becomes less efficient. Thus it should be straightforward to adapt our proof to models like [20] or [25], where we could however not formulate the security goal of entity authentication.

Thus we have chosen in essence the first model of Bellare and Rogaway [11] and the ability of the adversary to perform adaptive queries. Essentially equivalent variants of this model have been used by [15, 20], and especially by [47].

It is future work to adapt the adversarial capabilities to real attacks on browser based protocols, e.g. Cross Site Request Forgeries (CSRF) [61] enable an attacker to send Encrypt queries.