J Cryptol (2017) 30:1276–1324
DOI: 10.1007/s00145-016-9248-2

Journal of CRYPTOLOGY

# Authenticated Confidential Channel Establishment and the Security of TLS-DHE*

Tibor Jager
Department of Computer Science, Paderborn University, Paderborn, Germany
tibor.jager@upb.de

Florian Kohlar · Sven Schäge · Jörg Schwenk
Horst Görtz Institute for IT Security, Ruhr-University Bochum, Bochum, Germany
florian.kohlar@rub.de, sven.schaege@rub.de, joerg.schwenk@rub.de

Communicated by Hugo Krawczyk.

Received 6 November 2014 / Revised 5 November 2016
Online publication 18 January 2017

**Abstract.** Transport Layer Security (TLS) is the most important cryptographic protocol in use today. However, finding a cryptographic security proof for the complete, unaltered protocol has proven to be a challenging task. We give the first such proof in the standard model for the core cryptographic protocol underlying TLS cipher suites based on ephemeral Diffie–Hellman key exchange (TLS-DHE). This includes the cipher suite `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA`, which is mandatory in TLS 1.0 and TLS 1.1. It is impossible to prove the TLS Handshake secure in the classical security models of Bellare–Rogaway and Canetti–Krawczyk. The reason for this is that the final `Finished` messages of the TLS Handshake are encrypted with the session key, which provides an opportunity to distinguish real keys from random values. Therefore we start with proving the security of a truncated version of the TLS Handshake protocol, which has also been considered in previous work on TLS, and give the first proof of this variant in the standard model. Then we define the new notion of authenticated and confidential channel establishment (ACCE), which allows the monolithic analysis of protocols for which a modular security proof is not possible. We show that the combination of the TLS-DHE Handshake protocol and the TLS Record Layer encryption is secure in this model. Since the conference publication of this paper, the notion of ACCE has found many further applications, for example to the analysis of further TLS cipher suites (Krawczyk et al., Crypto 2013; Li et al., PKC 2014), advanced mechanisms like secure renegotiation of TLS session keys (Giesen et al., CCS 2013), and other practical protocols like EMV channel establishment (Brzuska et al., CCS 2013), SSH (Bergsma et al., CCS 2014), and QUIC (Lychev et al., S&P 2015).

**Keywords.** Authenticated key exchange, Authenticated confidential channel establishment (ACCE), SSL, TLS.

* This is an extended full version of a conference paper published at Crypto 2012 [52]. This work has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

## 1. Introduction

Transport Layer Security (TLS) is the most important Internet security mechanism today. Due to a subtle interleaving of the TLS Handshake protocol with the data encryption in the TLS Record Layer, it is impossible to prove the security of TLS using well-established security models [26,31,35], which define security via indistinguishability of keys. Prior to this work, there was no security proof for the complete protocol. Instead, all prior work either considered a modified version of the TLS Handshake [51,75], or weaker security goals [50].

In this paper, we provide new security results for the core cryptographic protocol of TLS based on ephemeral Diffie–Hellman key exchange (TLS-DHE). We give the first formal proof in the standard model that the truncated version of the TLS Handshake protocol, which has been subject to prior work on TLS [51,75], is an authenticated key exchange protocol in a security model that extends the Bellare–Rogaway model [26] to the public-key setting with adaptive corruptions and perfect forward secrecy (cf. [30]). Then we extend both the model and the proof to cover the combination of the TLS Handshake protocol with the TLS Record Layer, which allows us to show the security of the cryptographic core of a full TLS cipher suite.

In our analysis, we assume that the majority of building blocks of TLS-DHE (digital signature scheme, Diffie–Hellman key exchange, symmetric cipher) meets standard security properties. Solely for the pseudo-random function, we require an additional non-standard security assumption (PRF-ODH), which is a variant of the Oracle Diffie–Hellman assumption [1]. We also explain why such a non-standard assumptions seems hard to avoid, if a security model with corruptions is considered. Our proofs are stated for *mutual* authentication.

Proving Security of TLS 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` messages), which are essential to provide security against active adversaries, are first prepended with constant byte values (which provides us with known plaintext), then integrity protected by a MAC (which is instantiated with a pseudo-random function) and encrypted with the keys obtained from the TLS 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 plaintext. If this succeeds, he will output 'real', and otherwise 'random'. This makes it difficult to prove the full TLS Handshake protocol secure in any standard security model based on indistinguishability of keys. Morissey et al. [75] have therefore introduced a truncated TLS Handshake protocol, where the final encryption of the `Finished` messages is omitted, and have shown its security in the Random Oracle Model.

### 1.1. *Contributions*

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 security model for TLS 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, in the standard model. This allows to compare our results to previous work. The proof relies on the DDH assumption, an additional assumption called PRF-ODH, 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 analyse whether the building blocks have the required properties. Here we can partially build on previous work that analysed particular TLS components, see Sect. 1.2 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. In contrast to AKE protocols, where one requires *key indistinguishability*, we demand that a secure ACCE protocol allows to establish a 'secure communication channel' in the sense of stateful length-hiding authenticated encryption [81]. Loosely speaking, an ACCE channel guarantees that messages written to this channel are confidential (indistinguishable, and even the length of messages is concealed up to some granularity), that their integrity is preserved, and that a sequence of messages read from this channel corresponds exactly to the sequence of messages sent by the legitimate sender (of course up to dropping messages at the very end of the sequence, which is always possible). Since the conference publication of this paper, the notion of ACCE has found many further applications, for example to the analysis of further TLS cipher suites [61,64,72], advanced mechanisms like secure renegotiation of TLS session keys [49], and other practical protocols like EMV channel establishment [28], SSH [9], and QUIC [69,70], or post-quantum ACCE [86].

ACCE captures exactly the properties expected from TLS-like protocols. We prove that the core of the *full* TLS-DHE cipher suites, i.e. the combination of the TLS Handshake with the TLS Record Layer, forms a secure ACCE protocol, if the Record Layer provides security in the sense of stateful length-hiding authenticated encryption [81]. Note that CBC-based Record Layer protocols have been shown to provide length-hiding authenticated encryption by Paterson et al. [81].

Finally, we discuss the subtle property of TLS-DHE, which seems to make it hard to prove security without making an additional non-standard assumption, PRF-ODH. We also discuss several options to obtain a security proof using only standard assumptions, which include considering a weaker security model that disallows corruptions, and modifications to TLS-DHE.

Practical Impact Since the first publication of these results in 2012, support for TLS-DHE has increased significantly, since it is the only family of cipher suites providing perfect forward secrecy (PFS). Today more than 85% of all TLS servers support at least one TLS-DHE cipher suite, and the majority of them automatically switches to TLS-DHE if negotiating with a modern browser.[1] At the same time, mutual authentication is rarely used in practice. Typically, TLS is first used to authenticate the server and to establish a 'secure communication channel' between client and server. In the next step, the client authenticates itself by sending his authentication information over this secure communication channel to the server.

[1] SSL Pulse at https://trustworthyinternet.org, retrieved September 2016.

We believe that our result is nevertheless of practical value. First, the TLS-DHE-based cipher suite `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` is mandatory for TLS 1.0 and 1.1, which are both still in widespread use. Only the most recent version TLS 1.2 prescribes TLS with encrypted key transport as mandatory. In addition, the next version of TLS, currently discussed as TLS 1.3, will only support TLS-DHE-based cipher suites, albeit in a totally different handshake.

Second, in our analysis we can show that TLS-DHE provides *perfect forward secrecy*—a very strong form of long-term security, which basically requires that future compromises of long-term secrets do not threaten past communication sessions. With encrypted key transport, as in TLS-RSA, this is not achievable, since an attacker that compromises the long-term key (the private decryption key) can easily obtain session keys from previous sessions by just decrypting recorded encryptions of session keys. To better protect users from the consequences of such key-compromise attacks and offer better long-term security, service providers might therefore consider to switch to the (exclusive) use of TLS-DHE.

Interpretation Our results show that the core cryptographic protocol of TLS is cryptographically sound, if the building blocks are suitably secure. By combining our work with [81], we obtain a standard model security proof of core TLS 1.1 and 1.2 for *current* cipher suites if we assume directly that the signature scheme is EUF-CMA secure.[2]

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.

Choice of Security Model Authenticated key exchange is a basic building block in modern cryptography. However, since many different security models for different purposes exist [25,26,30–32,35,37,71], choosing 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. We have chosen an enhanced variant of the first model of Bellare and Rogaway [26]. Variants of this model have also been studied by [30,32], in particular by [75]. Detailed reasons for our choice are given in "Appendix A".

### 1.2. *Security Requirements on TLS Building Blocks*

In our proofs, we reduce the security of ephemeral Diffie–Hellman cipher suites to certain security properties of the building blocks of TLS. These building blocks (see Sect. 3 for precise definitions) are:

Pseudo-random function For the pseudo-random function used in TLS, we essentially require that (i) it is secure in the standard sense (see Definition 3) and (ii) it meets an additional requirement, PRF-ODH. The additional requirement on the PRF is related to the Oracle Diffie–Hellman (ODH) assumption, as introduced by Abdallah, Bellare, and

[2] To our best knowledge, there is no security proof for the currently used schemes, but also no result contradicting this assumption.

Rogaway in 2001 to prove security of DHIES [1]. Although the PRF-ODH assumption is non-standard, it is, in a way, the best we can achieve. We argue in Sect. 8 why we need this assumption for a proof in our security model. We also remark in Sect. 8 that we can obtain a proof under the DDH assumption instead of PRF-ODH, if we (a) either do not allow corruptions or (b) slightly modify the TLS Handshake. However, although the latter idea might guide future revisions of the TLS standard, it is not an option for the current work in which we are concerned with the security of the present TLS protocol. Disallowing corruptions on the other hand would make the security model unrealistically weak.

All TLS versions specify a HMAC-based construction of PRF that relies on different cryptographic hash functions. TLS 1.2 prescribes the use of SHA-256 [45], while previous standards used MD5 [84] and SHA-1 [46]. Foque et al. (Theorems 6,7 in [48]) were able to show that the pseudo-random function of TLS 1.2 constitutes a computationally strong randomness extractor for *two* different key 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)—the key may be either a random bit string, or a random element of a prime-order group (either a group defined over an elliptic curve or a *subgroup* of $Z_p^*$). 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 be obtained for TLS 1.0 and TLS 1.1. Complementing this, Fischlin, Lehmann, and Wagner showed that the same function also meets the strong security definition for pseudo-random functions [47] when used with a uniformly random key. In their analysis, Fischlin et al. relied on a result of Bellare [10] that shows the pseudo-randomness of HMAC.

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 communication channel' based on symmetric encryption (in the TLS Record Layer). While the informal idea of a 'secure communication channel' is simple, defining its security requirements precisely is not so trivial.

For instance, it is well known that using 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, e.g. it does not rule out replaying, dropping, or reordering of ciphertexts (cf. [21]). This issue can be solved by using a suitable *stateful* encryption scheme [17,18]. 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 information to an adversary, such as web browsing habits (e.g. [85]) or spoken phrases in Voice-over-IP connections (e.g. [88]). 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 introduced by Paterson et al. [81], as a formalization of the security properties that are expected from the TLS Record Layer encryption. The authors of [81] found that CBC-based Record Layer protocols of TLS 1.1 and 1.2 provably

meet this security goal under reasonable assumptions.[3] The results are not applicable to TLS 1.0, since it is well known that this version is insecure against chosen-plaintext attacks [5,6] since initialization vectors are not always chosen at random.

Digital signatures Our analysis furthermore requires that the employed signature scheme is secure in the standard sense of existential unforgeability under adaptive chosen-message attacks (see Definition 2). The current TLS standards offer three different signature schemes for authentication: RSASSA-PKCS#1 v1.5 [56], DSA [67], and ECDSA [53]. To our knowledge, currently there exists 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 [82,87].

### 1.3. *Previous Work on TLS*

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 [89]. Starting with the famous Bleichenbacher attack [19], many papers focus on various versions of the PKCS#1 standard [56] that defines the encryption padding used in TLS with RSA-encrypted key transport [34,51, 59,60]. At Crypto'02, Johnson and Kaliski showed that a simplified version of TLS with padded RSA is IND-CCA secure when modelling TLS as a 'tagged key-encapsulation mechanism' (TKEM) [51] under the strong non-standard assumption that a 'partial RSA decision oracle' is available.

In an independent line of research, several works analysed (simplified versions of) TLS using automated proof techniques in the Dolev–Yao model [44]. 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. Mitchell et al. used a finite-state enumeration tool named Murphi [74]. Ogata and Futatsugi used the interactive theorem prover OTS/CafeObj [79]. Paulson used the inductive method and the theorem prover Isabelle [80]. Unfortunately it is not known whether these proofs are actually cryptographically sound.

Bhargavan et al. [11] go two steps farther: first, they automatically derive their formal model from the source code of an TLS implementation, and second they try to automatize computational proofs using the CryptoVerif tool. Chaki and Datta [33] 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. [50] studied TLS in the Universal Composability (UC) framework [31]. The security definitions given in this paper (the 'ideal functionalities') are strictly weaker than the security guarantees we expect from TLS: for the Handshake part, only unauthenticated key exchange is modelled ($\mathcal{F}_{\mathrm{KE}}$), and thus the secure communication channel functionality ($\mathcal{F}_{\mathrm{SCS}}$) only guarantees confidentiality, not authenticity of endpoints. The paper further assumes that RSA-OAEP is used for encrypted key transport, which is not the case for current versions of TLS. Küsters and Tuengerthal [65]

[3] The proceedings version of [81] contains only a proof of *stateless* LHAE security. However, as also noted in [81], it is straightforward to adopt the results to the stateful setting.

claim to prove composable security for TLS assuming only local session identifiers, but leave out all details of the proof and only point to [50]. Moreover, Küsters and Tuengerthal point out that [50] also consider a modified TLS protocol, due to the need of unique session identifiers in UC.

Morissey et al. [75] analysed, in a paper that is closest to our results, the security of the truncated TLS Handshake protocol (cf. Sect. 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, and the order of messages of the TLS Handshake is slightly changed to better separate these three steps. They first consider a very weak class of passively secure key exchange 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 exchange of TLS. Again 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 fulfil 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 cipher suites of TLS secure in the standard model.

The work of Morissey et al. [75], which can be seen as a reference for the TLS Handshake protocol, considers also security of RSA-based cipher suites and thus is much broader in scope than our paper, but it does not cover our analysis of the TLS-DHE cipher suite. The modular proof strategy used in this paper is essentially bound to the random oracle model, since secure protocols for the premaster phase only yield secure protocols for the master phase if the master secret is derived from the premaster secret by evaluating a random oracle. Thus the ROM is used not only to allow a security proof for TLS-RSA cipher suites, but also to allow for a modular proof technique.

Paterson, Ristenpart, and Shrimpton [81] 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 cipher suites 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 [23,24] on authenticated encryption and on the analysis of different Mac-then-Encode-then-Encrypt (MEE) schemes analysed by Krawczyk [62] and Maurer and Tackmann [77].

### 1.4. *Subsequent Work on TLS and ACCE*

Applications of ACCE We propose the ACCE security model as a tool for the monolithic analysis of cryptographic protocols when a completely modular security analysis in simpler security models, like the BR [26] model or the CK [35] model, is not possible.

Subsequent to the publication of the conference version [52] of this paper, many works have adopted the ACCE model to prove security of other practical protocols that require a monolithic analysis. Some belong to the TLS family, while some are not directly related to TLS. In the sequel, we give an overview over recent developments in this direction.

Krawczyk, Paterson, and Wee [61] and concurrently Kohlar, Schäge, and Schwenk [64] used the ACCE model to provide security analyses for several other TLS cipher suites. To this end, they adopted the ACCE model to a setting where only the server is authenticated cryptographically (whereas we consider mutual cryptographic authentication). Most importantly, they are able to analyse the families of TLS-RSA and TLS-DH cipher suites. The work of Krawczyk et al. presents a refined analysis of (the cryptographic core of) TLS, which is more modular than our approach and that of the dedicated analyses of [64].

Further extending the coverage of various TLS cipher suites, Li et al. [72] give a security proof for TLS cipher suites with authentication via pre-shared keys (TLS-PSK). To this end, they extend the ACCE model to also cover authentication mechanisms with symmetric long-term secrets. To model PKI-related attacks, [64,72] extend the ACCE definition to also allow the adversary register new public (or pre-shared) keys.

The full TLS protocol suite is much more complex than the cryptographic core considered in this paper and in [61,64,72]. For instance, it additionally includes features like interactive agreement on a cipher suite, and mechanisms for renegotiation or abbreviated handshakes. Recently, the notion of ACCE security has also turned out useful for the analysis of protocols in such more complex settings. At CCS 2013, Giesen et al. [49] describe an extended ACCE model, which additionally includes a formal treatment of renegotiation in secure channel establishment protocols. Furthermore, Giesen et al. analyse the security of TLS with renegotiation, in particular the effectiveness of a countermeasure against the attack of Ray and Dispensa [83] employed in TLS.

The notion of ACCE security and adoptions of it to the specific needs of other practical protocols have also been used to analyse the security of constructions beyond TLS. At CCS 2013, Brzuska et al. [28] give a security proof for the channel establishment protocol of EMV. In a paper published at CCS 2014, Bergsma et al. [9] consider the security of the secure shell (SSH) protocol in a multi-cipher suite setting. Lychev, Jero, Boldyreva, and Nita-Rotar [69,70] provided a formal security analysis of Google's QUIC protocol, based on an ACCE variant for low-latency protocols called 'QACCE'.

One aspect of the relationship between AKE and ACCE was clarified by Brzuska et al. in [16]. The authors showed that secure *TLS-like* ACCE systems, i.e. systems that use some *master secret* to derive all subsequent keys, can be used to generate exportable session keys that are indistinguishable from random values.

Recent work on TLS (as a cryptographic protocol) A reference implementation of TLS 1.2, called miTLS, was presented by Bhargavan et al. [12], along with an automated verification of this implementation with the F7 typechecker. Their work allows to handle many advanced features of TLS, including full and abbreviated handshakes, but relies heavily on automation. In a different paper, Bhargavan et al. [13] use automated tools to analyse the miTLS reference implementation, including cipher

suite negotiation, key exchange, renegotiation, and resumption, and give a security proof based on the EasyCrypt [15] tool in combination with F7 typechecking and additional 'manual' proofs. An elaborate attack on authentication in TLS, which exploits a subtle combination of RSA and Diffie–Hellman cipher suites, session resumption, and session renegotiation (the *'Triple Handshake Attack'*), is described in [8]. A security analysis of TLS 1.2 and the recent draft of TLS 1.3 in the *constructive cryptography* framework is given in a recent paper by Kohlweiss et al. [57].

A different perspective on the PRF-ODH assumption is that this assumption essentially guarantees that the combination of Diffie–Hellman with the PRF in TLS forms a secure (one-time) key-encapsulation mechanism (KEM). This KEM-based view provides a more abstract and more modular view on this part of the TLS protocol. It was introduced and used in [13,61]. A nice feature of this more abstract view is that it makes the security analysis of TLS more modular and enables the analysis of TLS cipher suites beyond TLS-DHE, including TLS-RSA and TLS-DH.

Attacks on TLS It may seem strange that security proofs on TLS appear simultaneously with practical attacks on the protocol. This, however, does not contradict the proofs, since in each case one of the preconditions of the given security proofs has been violated, or a more complex scenario has been used.

In [2] and [3], a timing side channel from the Record Layer allows for data decryption. For CRIME [43], the length-hiding property of the Record Layer was broken. In POODLE [73], a padding oracle was used to break the Record Layer.

In [8], the combination of three different handshakes was shown to have some weaknesses. In [7] and [54], weaknesses in the implementations of some libraries were exploited, which failed to enforce the correct order of handshake messages, or the use of the correct EC group in cryptographic computations. In [39], it was assumed that the same RSA key pair was used in different protocol versions and for different purposes (encryption and signature verification).

TLS 1.3. The IETF is currently standardizing the successor of TLS 1.2, under the working title of TLS 1.3. It is a major revision of TLS and changes nearly every aspect of the protocol, from the handshake layout (only 1.5 RTT instead of 2 RTT for all previous TLS versions), the supported handshake families (only TLS-DHE will be supported), the Record Layer (a move to authenticated encryption), and even the security model: the main components of TLS 1.3 can be shown secure in classical AKE models.

The security of the novel handshake candidates has been studied in [39,40,58], the security of the novel Record Layer in [22,29]. A cross-version attack on TLS 1.3 has been described in [55]. A one-round key-exchange (ORKE)-based proposal as TLS 1.3 Handshake candidate, which had great influence on the standardization of Version 1.3, has been published in [66].

### 1.5. *Alternatives to ACCE*

Brzuska et al. [14] proposed relaxed game-based security notions for key exchange. This independent approach may serve as an alternative to ACCE to circumvent the impossibility of proving the TLS Handshake secure in a key-indistinguishability-based security model.

At Crypto 2014, Bhargavan et al. [13] presented the idea of modifying the security model in a different way in order to allow a security analysis of full TLS. Instead of including the record layer encryption into the model, as we do for ACCE, they introduce a novel non-standard *key-indistinguishability-based* security model. Essentially, the main novel idea behind their model is to release a real-or-random session key to the adversary *immediately* after the key is computed in the protocol. In particular, when TLS is considered in this model, then the key is released *before* the encrypted `Finished`-messages are sent. Bhargavan et al. [13] only consider the TLS Handshake without the encapsulating record layer. Therefore they essentially consider what we call 'truncated TLS'. However, interestingly and importantly the early key release idea seems to enable a generic composition of truncated TLS with a suitably secure record layer encryption scheme to establish the security of full TLS. The security definition of [13] focuses on the handshake and therefore does not define or prove security for record layer encryption. In contrast, the goal of ACCE is to capture all relevant security properties commonly expected from the cryptographic core of TLS.

## 2. Preliminaries and Definitions

We denote with $\emptyset$ the empty string and with $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ the set of integers from 1 to $n$. 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 we write $a \xleftarrow{\$} A$ to denote that $A$ is run with fresh random coins, producing output $a$.

### 2.1. *The Decisional Diffie–Hellman Assumption*

Let $G$ be a group of prime order $q$, and for $b \in \{0, 1\}$ let $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{DDH}}(b)$ denote the following security experiment.

1. Let $T_0 := (g, g^x, g^y, g^{xy})$ and $T_1 := (g, g^x, g^y, g^z)$ for $x, y, z \xleftarrow{\$} \mathbb{Z}_q$.
2. Run $b' \xleftarrow{\$} \mathcal{A}(T_b)$ and output $b'$.

**Definition 1.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{DDH}})$-*breaks* the DDH assumption in $G$, if $\mathcal{A}$ runs in time $t$ and

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{DDH}}(0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{DDH}}(1) = 1\right]\right| \geq \epsilon_{\mathsf{DDH}}$$

### 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 $\kappa$, 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 $\sigma$ 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 $(m, \sigma)$.

**Definition 2.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{SIG}})$-*breaks* the *existential unforgeability under adaptive chosen-message attacks* (EUF-CMA) of $\mathsf{SIG}$, if $\mathcal{A}$ runs in time $t$ and

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

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 and the PRFODH Assumption*

A *pseudo-random function* is an algorithm $\mathsf{PRF}$. 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 $z \in \{0, 1\}^\mu$. Let $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF}}(b)$ denote the following security experiment.

1. Let $F_0(\cdot) := \mathsf{PRF}(k, \cdot)$, where $k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ is a uniformly random key.
2. Let $F_1$ be a random function with the same domain and range as $\mathsf{PRF}$.
3. Run $b' \xleftarrow{\$} \mathcal{A}^{F_b}$. Whenever $\mathcal{A}$ queries an input value $x$ to $F_b$, respond with $z = F_b(x)$. Queries can be made adaptively. Finally, output $b'$.

**Definition 3.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{PRF}})$-*breaks* the security of $\mathsf{PRF}$, if it runs in time $t$ and

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF}}(0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF}}(1) = 1\right]\right| \geq \epsilon_{\mathsf{PRF}}$$

*Remark 1.* In 2008, Fouque et al. [48] showed that the HMAC-based key derivation function of TLS is a computationally strong randomness extractor for the source distributions $S_1$ and $S_2$ where (1) $S_1$ is a prime-order group of size $|S_1| = q$ that is either defined over an elliptic curve or as a subgroup of $\mathbb{Z}_p^*$ such that $q|p-1$, and (2) $S_2$ is the set of $l$-bitstrings $S_2 = \{0, 1\}^l$ and $l$ 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. Additionally, Fischlin et al. showed while relying on the pseudo-randomness of HMAC [10] that the pseudo-random function used in TLS is a pseudo-random function [47]. In the following, we will rely on the unifying assumption that the key derivation function in TLS is a pseudo-random function for key space $S_1$ and $S_2$.

Let $G$ be a group with generator $g$. Let $\mathsf{PRF}$ be a deterministic function $z = \mathsf{PRF}(X, m)$, taking as input a key $X \in G$ and some bit string $m$, and returning a string $z \in \{0,1\}^\mu$. Let $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF\text{-}ODH}}(b)$ be the following security experiment.

1. The adversary $\mathcal{A}$ outputs a value $m$.
2. The experiment samples $u, v \xleftarrow{\$} [q]$, $z_1 \xleftarrow{\$} \{0,1\}^\mu$ uniformly random and sets $z_0 := \mathsf{PRF}(g^{uv}, m)$. Then it returns $z_b$, $g^u$ and $g^v$ to the adversary.
3. The adversary may query a pair $(X, m')$ with $X \neq g^u$ to the experiment. The experiment replies with $\mathsf{PRF}(X^v, m')$.
4. Finally, the adversary outputs a guess $b' \in \{0,1\}$. The experiment outputs $b'$.

**Definition 4.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{PRF\text{-}ODH}})$-*breaks* the $\mathsf{PRF\text{-}ODH}$ assumption with respect to $G$ and $\mathsf{PRF}$, if it runs in time $t$ and it holds that

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF\text{-}ODH}}(0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{PRF\text{-}ODH}}(1) = 1\right]\right| \geq \epsilon_{\mathsf{PRF\text{-}ODH}}$$

The PRF-Oracle-Diffie–Hellman ($\mathsf{PRF\text{-}ODH}$) assumption is a variant of the $\mathsf{ODH}$ assumption introduced by Abdalla, Bellare, and Rogaway in [1], adopted from hash functions to PRFs. In contrast to allowing a polynomial number of queries as in the original assumption [1], we allow only a single oracle query.

### 2.4. *Collision-Resistant Hashing*

**Definition 5.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{H}})$-*breaks the collision resistance* of hash function $\mathsf{H}$, if it runs in time $t$ and it holds that

$$\Pr\left[\mathcal{A}(\mathsf{H}) = (m, m') : m \neq m' \wedge \mathsf{H}(m) = \mathsf{H}(m')\right] \geq \epsilon_{\mathsf{H}}$$

### 2.5. *Stateful Length-Hiding Authenticated Encryption*

Let us now describe the stateful variant of LHAE security. The following description and security model were obtained from the authors of [81] via personal communication, see [81] for a detailed discussion and motivation of this security model.

A *stateful length-hiding symmetric encryption scheme* consists of three algorithms $\mathsf{StE} = (\mathsf{StE.Init}, \mathsf{StE.Enc}, \mathsf{StE.Dec})$. Algorithm $(st_e, st_d) = \mathsf{StE.Init}()$ initializes all states used by the encryption scheme. Algorithm $(C, st'_e) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, H, m, st_e)$ takes as input a secret key $k \in \{0,1\}^\kappa$, an output ciphertext length $\mathsf{len} \in \mathbb{N}$, some header data $H \in \{0,1\}^*$, a plaintext $m \in \{0,1\}^*$, and the current state $st_e \in \{0,1\}^*$, and outputs either a ciphertext $C \in \{0,1\}^{\mathsf{len}}$ and an updated state $st'_e$ or an error symbol $\perp$.[4] 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 an error symbol $\perp$ indicating that $C$ is not a valid ciphertext.

[4] For instance, if the output length $\mathsf{len}$ is smaller than the length of message $m$.

| $\underline{\mathsf{Encrypt}(m_0, m_1, \mathsf{len}, H)}$: | $\underline{\mathsf{Decrypt}(C, H)}$: |
|---|---|
| $u := u + 1$ | $v := v + 1$ |
| $(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, H, m_0, st_e)$ | If $b = 0$, then return $\perp$ |
| $(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, H, m_1, st_e)$ | $(m, st_d) = \mathsf{StE.Dec}(k, H, C, st_d)$ |
| If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return $\perp$ | If $v > u$ or $C \neq C_v$ or $H \neq H_v$, then $\mathsf{phase} := 1$ |
| $(C_u, st_e) := (C^{(b)}, st_e^{(b)})$ | If $\mathsf{phase} = 1$ then return $m$ |
| Return $C_u$ | Return $\perp$ |

**Fig. 1.** Encrypt and Decrypt oracles in the stateful LHAE security experiment.

For $b \in \{0, 1\}$, let $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{sLHAE}}(b)$ denote the following experiment.

- Choose $k \xleftarrow{\$} \{0, 1\}^{\kappa}$, and set $(st_e, st_d) \xleftarrow{\$} \mathsf{StE.Init}$.
- Run $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}$ and output $b'$.

Here $\mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}$ denotes that $\mathcal{A}$ has access to two oracles Encrypt and Decrypt. The encryption oracle $\mathsf{Encrypt}(m_0, m_1, \mathsf{len}, H)$ takes as input two messages $m_0$ and $m_1$, length parameter len, and header data $H$. It maintains a counter $u$ which is initialized to 0. Oracle $\mathsf{Decrypt}(C, H)$ takes as input a ciphertext $C$ and header $H$ and keeps a counter $v$ and a variable phase; both are initialized to 0. Both oracles process a query as defined in Fig. 1.

**Definition 6.** We say that adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{sLHAE}})$-breaks the stateful symmetric encryption scheme, if $\mathcal{A}$ runs in time $t$ and

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{sLHAE}}(0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{sLHAE}}(1) = 1\right]\right| \geq \epsilon_{\mathsf{sLHAE}}$$

*Remark 2.* The sequence numbers used in the computation of the TLS Record Layer MACs are contained, for each direction, in the state information $st_e$ and $st_d$. If they do not match, decryption will fail. The counters $u$ and $v$ are additional counters needed to define the security experiment.

## 3. Transport Layer Security

The current version of TLS is 1.2 [42]; it coexists with its predecessors TLS 1.0 [38] and TLS 1.1 [41]. In the following, we give a description of all messages sent during the TLS Handshake with ephemeral Diffie–Hellman key exchange and client authentication (i.e. for cipher suites `TLS_DHE_*`). This description and its illustration in Fig. 2 are valid for all TLS versions from 1.0 to 1.2. Our description makes use of several 'state variables' $(\Lambda, k, \Pi, \rho, st)$. For instance, variable $\Lambda \in \{\texttt{accept}, \texttt{reject}\}$ determines whether one party 'accepts' or 'rejects' an execution of the protocol, or variable $k$ stores the session key. These variables will also appear later in our security model (Sect. 4).

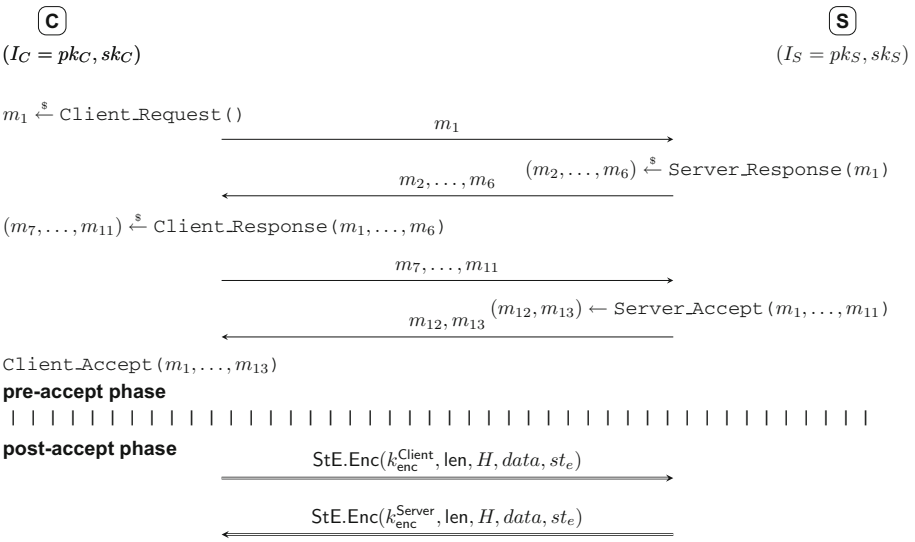


**Fig. 2.** TLS Handshake for cipher suites `TLS_DHE_*` with client authentication.

The TLS-DHE 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 start-up provides no encryption nor any other cryptographic transformations (Fig. 3).

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. This is followed by a list `cs-list` of *cipher suites*, where each cipher suite is a tuple of key exchange method, signing, encryption, and MAC algorithms, coded as two bytes. Data compression is possible before encryption and is signalled by the inclusion of zero or more compression methods.

Server Hello and Server Key Exchange The `Server Hello` message $m_2$ has the same structure as `Client Hello`, with the only exception that at most one cipher suite 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. When the certificate $cert_S$ is received, the client sets its partner id $\Pi{:=}S$. The public key in the certificate must match the cipher suite 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)

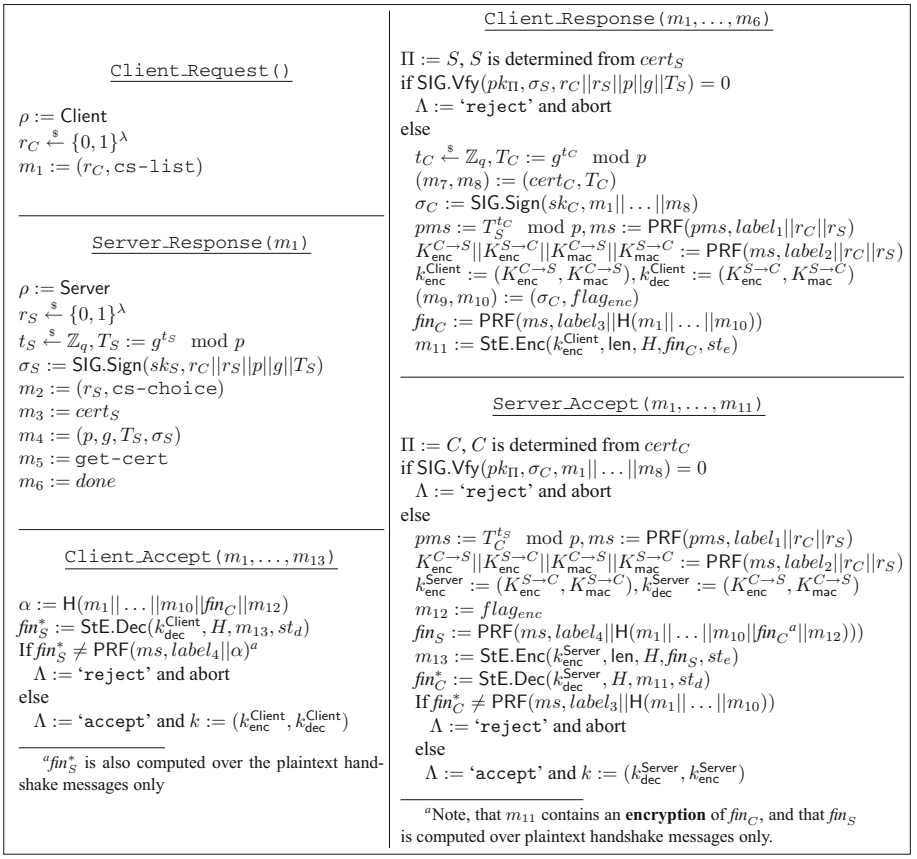


**Fig. 3.** Computation of client/server handshake messages.

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 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 and Client Finished Having received these messages, the signature $\sigma_S$ is verified. If this fails, the client 'rejects' and aborts. Otherwise, after successful verification, the client is able to complete the key exchange 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. When the certificate $cert_C$ is received by the server, the server sets its partner id $\Pi{:=}C$. To authenticate the client, a signature $\sigma_C$ 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 `Client Finished` message $fin_C$. More precisely, the key material $k_{\mathsf{enc}}^{\mathsf{Client}}:=(K_{\mathsf{enc}}^{C\to S}, K_{\mathsf{mac}}^{C\to S})$ and $k_{\mathsf{dec}}^{\mathsf{Client}}:=(K_{\mathsf{enc}}^{S\to C}, K_{\mathsf{mac}}^{S\to C})$ is computed as

$$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C}:=\mathsf{PRF}(ms, label_2||r_C||r_S) \tag{1}$$

where $k_{\mathsf{enc}}^{\mathsf{Client}}$ is used to encrypt and authenticate data sent from the client to the server and $k_{\mathsf{dec}}^{\mathsf{Client}}$ is used to decrypt and verify data received from the server.

After these computations have been completed, the keys are handed over to the TLS Record Layer of the client, which is now able to encrypt and MAC 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 $S$. Then message $m_{11}$ consists of an authenticated encryption of the `Client Finished` message $fin_C$ which is computed as

$$fin_C:=\mathsf{PRF}(ms, label_3||\mathsf{H}(m_1||\ldots||m_{10}))$$

where $\mathsf{H}$ is a hash function specified by the negotiated cipher suite.

*Remark 3.* Please note that a padding is applied to $fin_C$ before encryption and that this padding allows for (partially) known plaintext attacks on $m_{11}$. Thus if we analyse TLS in one of the classical key-indistinguishability-based security models of [26] or [35], then the answer to a `Test` query could be determined by simply decrypting $m_{11}$, and checking if the resulting plaintext has the appropriate padding. An alternative approach, based on the idea of 'early key release', was described after the conference publication of this paper in [13].

SERVER FINISHED After the server has received messages $m_7$, $m_8$, $m_9$, the server verifies the signature in $m_9$. If this fails, the server 'rejects' (i.e. sets $\Lambda =$ '`reject`') and aborts. Otherwise it first determines $pms$ and $ms$. From this the encryption and MAC keys $k_{\mathsf{enc}}^{\mathsf{Server}}:=(K_{\mathsf{enc}}^{S\to C}, K_{\mathsf{mac}}^{S\to C})$ and $k_{\mathsf{dec}}^{\mathsf{Server}}:=(K_{\mathsf{enc}}^{C\to S}, K_{\mathsf{mac}}^{C\to S})$ are computed as in (1).[5] It 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, it 'rejects' and aborts. If the check is successful, it 'accepts' (i.e. sets $\Lambda =$ '`accept`') and computes the `Server Finished` message $fin_S$ over all **plaintext** handshake messages as

$$fin_S:=\mathsf{PRF}(ms, label_4||\mathsf{H}(m_1||\ldots||m_{10}||fin_C||m_{12}))$$

Then it sends messages $m_{12}$ ($flag_{enc}$) and $m_{13}$ (the encryption of $fin_S$) to the client. If the check of $fin_S$ on the client side is successful, the client also 'accepts'.

[5] Note that we have $k_{\mathsf{enc}}^{\mathsf{Server}} = k_{\mathsf{dec}}^{\mathsf{Client}}$ and $k_{\mathsf{dec}}^{\mathsf{Server}} = k_{\mathsf{enc}}^{\mathsf{Client}}$.

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. Sect. 2.5). 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 $\mathsf{len}$ and encrypted ('MAC-then-Encode-then-Encrypt').

The state $st_d$ of the decryption algorithm consists of a sequence number, which is incremented on each decryption operation. Given a ciphertext, the algorithm decrypts and verifies the MAC using its own sequence counter.

*Remark 4.* Our security analysis is not based on any specific details of the CBC-based record layer protocol. We will only require that the record layer encryption scheme is sLHAE-secure. Thus, the analysis applies to other record layer encryption schemes as well, provided that it is possible to prove (or at least reasonable to assume) that it is sLHAE-secure.

Abbreviated TLS Handshakes, side channels, and cross-protocol attacks In our analysis, we do not consider abbreviated TLS Handshakes, but we note that the server can always enforce a full TLS Handshake. Moreover, we do not consider attacks based on side channels, such as error messages or implementation issues, or cross-cipher suite or cross-protocol attacks [78,89]. Please note that cross-version attacks as proposed in [55] may have real-world impact, as the example of the DROWN attack has shown [4].

## 4. AKE Protocols

While the established security models for, say, encryption (e.g. IND-CPA or IND-CCA security), or digital signatures (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 [30,35,37,71] dates back to Bellare and Rogaway [26], 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. In the sequel, we describe a variant of this model, which captures adaptive corruptions, perfect forward secrecy, and security against key-compromise impersonation attacks in a public-key setting.

### 4.1. *Execution Environment*

Consider a set of parties $\{P_1, \ldots, P_\ell\}$, $\ell \in \mathbb{N}$, where each party $P_i \in \{P_1, \ldots, P_\ell\}$ is a (potential) protocol participant and has a long-term key pair $(pk_i, sk_i)$. To model several sequential and parallel executions of the protocol, each party $P_i$ is modelled

by a collection of oracles $\pi_i^1, \ldots, \pi_i^d$ for $d \in \mathbb{N}$. Each oracle $\pi_i^s$ represents a process that executes one single 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 $\pi_i^s$ maintains as internal state the following variables:

- $\Lambda \in \{\texttt{accept}, \texttt{reject}\}$.
- $k \in \mathcal{K}$, where $\mathcal{K}$ is the keyspace of the protocol.
- $\Pi \in \{1, \ldots, \ell\}$ containing the intended communication partner, i.e. an index that points to a public key $pk_\Pi$ used to perform authentication within the protocol execution.[6]
- Variable $\rho \in \{\mathsf{Client}, \mathsf{Server}\}$.
- 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 TLS Handshake).

The internal state of each oracle is initialized to $(\Lambda, k, \Pi, \rho, st) = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, where $V = \emptyset$ denotes that variable $V$ is undefined. 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

$$k \neq \emptyset \iff \Lambda = \texttt{accept} \tag{2}$$

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 message $m$ to oracle $\pi_i^s$. The oracle will respond according to the protocol specification, depending on its internal state.
  If the adversary asks the first Send-query to oracle $\pi_i^s$, then the oracle checks whether $m = \top$ consists of a special 'initialization' symbol $\top$. If true, then it sets its internal variable $\rho$:=Client and responds with the first protocol message. Otherwise it sets $\rho$:=Server and responds as specified in the protocol.[7]
  The variables $\Lambda, k, \Pi, st$ are also set after a Send-query. When and how depends on the considered protocol.
- $\mathsf{Reveal}(\pi_i^s)$: Oracle $\pi_i^s$ responds to a Reveal-query with the contents of variable $k$. Note that we have $k \neq \emptyset$ if and only if $\Lambda =$ '`accept`', see (2).
- $\mathsf{Corrupt}(P_i)$: Oracle $\pi_i^1$ responds with the long-term secret key $sk_i$ of party $P_i$.[8] If $\mathsf{Corrupt}(P_i)$ is the $\tau$th query issued by $\mathcal{A}$, then we say that $P_i$ is $\tau$-*corrupted*. For parties that are not corrupted, we define $\tau := \infty$.

[6] We assume that each party $P_i$ is uniquely identified by its public key $pk_i$. In practice, several keys may be assigned to one identity. Furthermore, there may be other ways to determine identities, for instance by using certificates. However, this is out of scope of this paper.

[7] Note that we do not include the identity of the (intended) communication partner in the Send-query. Instead, we assume that the exchange of identities of communication partners (which is necessary to determine the public key used to perform authentication) is part of the protocol.

[8] Note that the adversary does not 'take control' of oracles corresponding to a corrupted party. But he learns the long-term secret key and can henceforth simulate these oracles. Still, corrupted oracles remain functional, which is necessary to capture security against KCI attacks.

- $\mathsf{Test}(\pi_i^s)$: This query may be asked only once throughout the game. If $\pi_i^s$ has state $\Lambda \neq \texttt{accept}$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin $b$, samples an independent key $k_0 \xleftarrow{\$} \mathcal{K}$, sets $k_1 = k$ to the 'real' key computed by $\pi_i^s$, and returns $k_b$.

The Send-query 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 Corrupt-query allows the adversary to learn $sk_i$ of party $P_i$, it may for instance be used by $\mathcal{A}$ to impersonate $P_i$. The Test-query will be used to define security.

### 4.2. *Security Definition*

Bellare and Rogaway [26] have introduced the notion of *matching conversations* in order to define correctness and security of an AKE protocol precisely.

We denote with $T_{i,s}$ the sequence that consists of all messages sent and received by $\pi_i^s$ in chronological order (not including the initialization-symbol $\top$). We also say that $T_{i,s}$ is the *transcript* of $\pi_i^s$. For two transcripts $T_{i,s}$ and $T_{j,t}$, we say that $T_{i,s}$ is a *prefix* of $T_{j,t}$, if $T_{i,s}$ contains at least one message, and the messages in $T_{i,s}$ are identical to and in the same order as the first $|T_{i,s}|$ messages of $T_{j,t}$.

**Definition 7.** (Matching conversations) We say that $\pi_i^s$ has a *matching conversation* to $\pi_j^t$, if

- $T_{j,t}$ is a prefix of $T_{i,s}$ and $\pi_i^s$ has sent the last message(s), or
- $T_{i,s} = T_{j,t}$ and $\pi_j^t$ has sent the last message(s).

*Remark 5.* 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 are sent sequentially. For instance, in the TLS Handshake protocol (see Fig. 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 the presence of an active adversary that simply drops the last protocol message.

Security of AKE protocols is now defined by requiring that (i) the protocol is a secure authentication protocol 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 $\ell$ 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. Now the adversary may start issuing Send, Reveal and Corrupt queries, as well as one Test-query. Finally, the adversary outputs a bit $b'$ and terminates.

**Definition 8.** Assume a 'benign' adversary $\mathcal{A}$, which picks two arbitrary oracles $\pi_i^s$ and $\pi_j^t$ and performs a sequence of Send-queries by faithfully forwarding all messages between $\pi_i^s$ and $\pi_j^t$. Let $k_i^s$ denote the key computed by $\pi_i^s$ and let $k_j^t$ denote the key computed by $\pi_j^t$. We say that an AKE protocol is *correct*, if for this benign adversary and any two oracles $\pi_i^s$ and $\pi_j^t$ always holds that

1. both oracles have $\Lambda = \texttt{accept}$,[9] and
2. $k_i^s = k_j^t \in \mathcal{K}$.

**Definition 9.** We say that an adversary $(t, \epsilon)$-*breaks* an AKE protocol $\Phi$, if $\mathcal{A}$ runs in time $t$, and at least one of the following two conditions holds:

1. When $\mathcal{A}$ terminates, then with probability at least $\epsilon$ there exists an oracle $\pi_i^s$ such that
   - $\pi_i^s$ 'accepts' when $\mathcal{A}$ issues its $\tau_0$th query with intended partner $\Pi = j$, and
   - $P_j$ is $\tau_j$-corrupted with $\tau_0 < \tau_j$,[10] and
   - there is no unique oracle $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$.

   If an oracle $\pi_i^s$ accepts in the above sense, then we say that $\pi_i^s$ accepts *maliciously*.
2. When $\mathcal{A}$ issues a Test-query to any oracle $\pi_i^s$ and
   - $\pi_i^s$ 'accepts' when $\mathcal{A}$ issues its $\tau_0$th query with intended partner $\Pi = j$, and
   - $P_j$ is $\tau_j$-corrupted with $\tau_0 < \tau_j$, and
   - $\mathcal{A}$ does not issue a Reveal-query to $\pi_i^s$, nor to $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$ (if such an oracle exists),

   then the probability that $\mathcal{A}$ outputs $b'$ which equals the bit $b$ sampled by the Test-query satisfies

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

   If an adversary $\mathcal{A}$ outputs $b'$ such that $b' = b$ and the above conditions are met, then we say that $\mathcal{A}$ *answers the* Test*-challenge correctly*.

*Remark 6.* Note that the above definition even allows to corrupt oracles involved in the Test-session (of course only after the Test-oracle has reached `accept`-state, in order to exclude trivial attacks). Thus, protocols secure with respect to this definition provide *perfect forward secrecy*. Note also that we allow the 'accepting' oracle to be corrupted even *before* it reaches `accept`-state, which provides security against *key-compromise impersonation* attacks.

[9] We do not demand that partner ids $\Pi$ are mutually matching. However, this is required by the security definition.

[10] That is, $P_j$ is not corrupted when $\pi_i^s$ 'accepts'. Recall that uncorrupted parties are $\tau$-corrupted with $\tau = \infty$.

## 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. We consider a 'truncated TLS' protocol as in [75,76]. 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 Fig. 2 such that

- message $m_{11}$ contains only $fin_C$ (instead of $\mathsf{StE.Enc}(k_{\mathsf{enc}}^{\mathsf{Client}}, \mathsf{len}, H, fin_C, st_e)$), and
- message $m_{13}$ contains only $fin_S$ (instead of $\mathsf{StE.Enc}(k_{\mathsf{enc}}^{\mathsf{Server}}, \mathsf{len}, H, fin_S, st_e)$).

This simple modification allows to prove security in the key-indistinguishability-based security model from Sect. 4.

In the following, we will consider three types of adversaries:

1. Adversaries that make an oracle accept maliciously (in the sense of Definition 9), such that the first oracle that does so is a Client-oracle (i.e. an oracle with $\rho = \mathsf{Client}$). We call such an adversary a Client-adversary.
2. Adversaries that make an oracle accept maliciously (in the sense of Definition 9), such that the first oracle that does so is a Server-oracle (i.e. an oracle with $\rho = \mathsf{Server}$). We call such an adversary a Server-adversary.
3. Adversaries that do not make any oracle accept maliciously (in the sense of Definition 9). We call such an adversary a Test-adversary.

Note that any adversary, which is successful in the sense of Definition 9, is either a Client-adversary, or a Server-adversary, or a Test-adversary.

**Theorem 1.** *From any adversary $\mathcal{A}$ that $(t', \epsilon_{\mathsf{ttls}})$-breaks the truncated ephemeral Diffie–Hellman TLS Handshake protocol in the sense of Definition 9, we can construct an adversary $\mathcal{A}_{\mathsf{PRF}}$ that $(t, \epsilon_{\mathsf{PRF}})$-breaks the security of PRF, $\mathcal{A}_{\mathsf{sig}}$ that $(t, \epsilon_{\mathsf{sig}})$-breaks the security of the signature scheme, $\mathcal{A}_{\mathsf{ddh}}$ that $(t, \epsilon_{\mathsf{DDH}})$-breaks the DDH assumption in the group $G$ used to compute the TLS-DHE premaster secret, $\mathcal{A}_{\mathsf{H}}$ that $(t, \epsilon_{\mathsf{H}})$-breaks the collision resistance of H, and $\mathcal{A}_{\mathsf{prfodh}}$ that $(t, \epsilon_{\mathsf{PRF\text{-}ODH}})$-breaks the PRF-ODH-problem with respect to $G$ and PRF, with $t \approx t'$ and the following lower bounds on the success probabilities of the constructed adversaries.*

- *If $\mathcal{A}$ is a Client-adversary, then it holds that*

$$\epsilon_{\mathsf{ttls}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + d\ell \left( \epsilon_{\mathsf{PRF\text{-}ODH}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \frac{1}{2^{\mu}} \right) \right)$$

- *If $\mathcal{A}$ is a Server-adversary, then it holds that*

$$\epsilon_{\mathsf{ttls}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \frac{1}{2^{\mu}} \right)$$

- *If $\mathcal{A}$ is a Test-adversary, then it holds that*

$$\epsilon_{\mathsf{ttls}} \leq d\ell \cdot (\epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}})$$

*Recall here that $\ell$ denotes the number of parties in the security model, $d$ the number of sessions per party, $\mu$ the length of the* FINISHED *messages $fin_S$ and $fin_C$, and $\lambda$ the length of the nonces $r_C$ and $r_S$.*

Theorem 1 is proven by Lemmas 1, 2, and 3 given as follows.

### 5.1. *Authentication*

**Lemma 1.** *From any* Client*-adversary $\mathcal{A}$ that runs in time $t'$ with success probability $\epsilon_{\mathsf{client}}$, we can construct adversaries $\mathcal{A}_{\mathsf{sig}}$, $\mathcal{A}_{\mathsf{prfodh}}$, $\mathcal{A}_{\mathsf{PRF}}$, and $\mathcal{A}_{\mathsf{H}}$ as in Theorem 1, with*

$$\epsilon_{\mathsf{client}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + d\ell \left( \epsilon_{\mathsf{PRF\text{-}ODH}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \frac{1}{2^{\mu}} \right) \right)$$

*where all quantities are defined as in Theorem 1.*

*Proof.* The proof proceeds in a *sequence of games*, following [27]. The first game is the real security experiment. We then describe several intermediate games that modify the original game step by step, each time bounding the difference in the success probability of the adversary between each two successive games. We end up in the final game, where no adversary can break the security of the protocol.

Let $\mathsf{break}_{\delta}^{(1)}$ be the event that occurs when the first oracle accepts maliciously in the sense of Definition 9 with $\rho = \mathsf{Client}$ in Game $\delta$.

**Game 0** This game equals the AKE security experiment described in Sect. 4. Thus, for some $\epsilon_{\mathsf{client}}$ we have

$$\Pr\left[\mathsf{break}_0^{(1)}\right] = \epsilon_{\mathsf{client}}$$

**Game 1** In this game, we add an abort rule. The challenger aborts, if there exists any oracle $\pi_i^s$ that chooses a random nonce $r_C$ or $r_S$ which is not unique. More precisely, the game is aborted if the adversary ever makes a first Send query to an oracle $\pi_i^s$, and the oracle replies with random nonce $r_C$ or $r_S$ such that there exists some other oracle $\pi_{i'}^{s'}$ which has previously sampled the same nonce.

In total less than $d\ell$ nonces $r_C$ and $r_S$ are sampled, each uniformly random from $\{0,1\}^{\lambda}$. Thus, the probability that a collision occurs is bounded by $(d\ell)^2 2^{-\lambda}$, which implies

$$\Pr\left[\mathsf{break}_0^{(2)}\right] \leq \Pr\left[\mathsf{break}_1^{(2)}\right] + \frac{(d\ell)^2}{2^{\lambda}}$$

Note that now each oracle has a unique nonce $r_C$ or $r_S$, which is included in the signatures. We will use this to ensure that each oracle that accepts with non-corrupted partner has a *unique* partner oracle.

**Game 2** We try to guess which client-oracle will be the first oracle to accept maliciously. If our guess is wrong, i.e. if there is another (Client or Server) oracle that accepts maliciously earlier, then we abort.

Technically, this game is identical to Game 1, except for the following. The challenger guesses two random indices $(i^*, s^*) \xleftarrow{\$} [\ell]\times[d]$. If there exists an oracle $\pi_i^s$ that 'accepts' maliciously, and $(i, s) \neq (i^*, s^*)$ and $\pi_i^s$ has $\rho \neq \mathsf{Client}$, then the challenger aborts the game. Note that if the first oracle $\pi_i^s$ that 'accepts' maliciously has $\rho = \mathsf{Client}$, then with probability $1/(d\ell)$ we have $(i, s) = (i^*, s^*)$, and thus

$$\Pr\left[\mathsf{break}_1^{(2)}\right] = d\ell \cdot \Pr\left[\mathsf{break}_2^{(2)}\right]$$

Note that in this game the adversary can only break the security of the protocol, if oracle $\pi_{i*}^{s^*}$ is the first oracle that 'accepts' maliciously and has $\rho = \mathsf{Client}$, as otherwise the game is aborted.

**Game 3** Again the challenger proceeds as before, but we add an abort rule. We want to make sure that $\pi_{i*}^{s^*}$ receives as input exactly the Diffie–Hellman value $T_S$ that was selected by some other uncorrupted oracle that received the nonce $r_C$ chosen by $\pi_{i*}^{s^*}$ as first input (note that there may be several such oracles, since the adversary may send copies of $r_C$ to many oracles).

Technically, we abort and raise event $\mathsf{abort}_{\mathsf{sig}}$, if oracle $\pi_{i*}^{s^*}$ ever receives as input a message $m_3 = cert_S$ indicating intended partner $\Pi = j$ and message $m_4 = (p, g, T_S, \sigma_S)$ such that $\sigma_S$ is a valid signature over $r_C||r_S||p||g||T_S$, but there exists no oracle $\pi_j^t$ which has previously output $\sigma_S$. Clearly we have

$$\Pr\left[\mathsf{break}_2^{(1)}\right] \leq \Pr\left[\mathsf{break}_3^{(1)}\right] + \Pr[\mathsf{abort}_{\mathsf{sig}}]$$

Note that the experiment is aborted, if $\pi_{i*}^{s^*}$ does not accept *maliciously*, due to Game 2. This means that party $P_j$ must be $\tau_j$-corrupted with $\tau_j = \infty$ (i.e. not corrupted) when $\pi_{i*}^{s^*}$ accepts (as otherwise $\pi_{i*}^{s^*}$ does not accept *maliciously*). To show that $\Pr[\mathsf{abort}_{\mathsf{sig}}] \leq \ell \cdot \epsilon_{\mathsf{sig}}$, we construct a signature forger $\mathcal{A}_{\mathsf{sig}}$ 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 3, except that it uses its chosen-message oracle to generate a signature under $pk_\phi$ when necessary.

If $\phi = j$, which happens with probability $1/\ell$, then the forger can use the signature received by $\pi_{i*}^{s^*}$ to break the EUF-CMA security of the signature scheme with success probability $\epsilon_{\mathsf{sig}}$, so $\Pr[\mathsf{abort}_{\mathsf{sig}}]/\ell \leq \epsilon_{\mathsf{sig}}$. Therefore we have

$$\Pr\left[\mathsf{break}_2^{(1)}\right] \leq \Pr\left[\mathsf{break}_3^{(1)}\right] + \ell \cdot \epsilon_{\mathsf{sig}}$$

Note that in Game 3 oracle $\pi_{i*}^{s^*}$ receives as input a Diffie–Hellman value $T_S$ such that $T_S$ was chosen by another oracle, but not by the adversary. Note also that there may be

multiple oracles that issued a signature $\sigma_S$ containing $r_C$, since the adversary may have sent several copies of $r_C$ to several oracles.

**Game 4** In this game, we want to make sure that we know *which* oracle $\pi_j^t$ will issue the signature $\sigma_S$ that $\pi_{i*}^{s^*}$ receives. Note that this signature includes the random nonce $r_S$, which is unique due to Game 1. Therefore the challenger in this game proceeds as before, but additionally guesses two indices $(j^*, t^*) \xleftarrow{\$} [\ell] \times [d]$. It aborts, if the adversary does *not* make a $\mathsf{Send}$-query containing $r_C$ to $\pi_{j^*}^{t^*}$, and $\pi_{j^*}^{t^*}$ responds with messages containing $\sigma_S$ such that $\sigma_S$ is forwarded to $\pi_{i*}^{s^*}$.

We know that there must exist at least one oracle that outputs $\sigma_S$ such that $\sigma_S$ is forwarded to $\pi_{i*}^{s^*}$, due to Game 3. Thus we have

$$\Pr\left[\mathsf{break}_3^{(1)}\right] \leq d\ell \cdot \Pr\left[\mathsf{break}_4^{(1)}\right]$$

Note that in this game we know exactly that oracle $\pi_{j*}^{t^*}$ chooses the Diffie–Hellman share $T_S$ that $\pi_{i*}^{s^*}$ uses to compute its premaster secret.

**Game 5** Recall that $\pi_{i*}^{s^*}$ computes the master secret as $ms = \mathsf{PRF}(T_S^{t_c}, label_1||r_C||r_S)$, where $T_S$ denotes the Diffie–Hellman share received from $\pi_{j*}^{t^*}$ and $t_c$ denotes the Diffie–Hellman exponent chosen by $\pi_{i*}^{s^*}$. In this game, we replace the master secret $ms$ computed by $\pi_{i*}^{s^*}$ with an independent random value $\widetilde{ms}$. Moreover, if $\pi_{j*}^{t^*}$ receives as input the same Diffie–Hellman share $T_C$ that was sent from $\pi_{i*}^{s^*}$, then we set the master secret of $\pi_{j*}^{t^*}$ equal to $\widetilde{ms}$. Otherwise we compute the master secret as specified in the protocol.

Suppose there exists an adversary $\mathcal{A}$ that distinguishes Game 5 from Game 4. We show that this implies an adversary $\mathcal{A}_{\mathsf{prfodh}}$ that breaks the $\mathsf{PRF\text{-}ODH}$ assumption. Adversary $\mathcal{A}_{\mathsf{prfodh}}$ outputs $(label_1||r_C||r_S)$ to its experiment and receives in response $(g, g^u, g^v, R)$, where either $R = \mathsf{PRF}(g^{uv}, label_1||r_C||r_S)$ or $R \xleftarrow{\$} \{0,1\}^\mu$. It runs $\mathcal{A}$ by implementing the challenger for $\mathcal{A}$ and embeds $(g^u, g^v)$ as follows. Instead of letting $\pi_{i*}^{s^*}$ choose $T_C = g^{t_C}$ for random $t_C \xleftarrow{\$} \mathbb{Z}_q$, $\mathcal{A}_{\mathsf{prfodh}}$ defines $T_C{:=}g^u$. Similarly, the Diffie–Hellman share $T_S$ of $\pi_{j*}^{t^*}$ is defined as $T_S{:=}g^v$. Finally, the master secret of $\pi_{i*}^{s^*}$ is set equal to $R$.

Note that $\pi_{i*}^{s^*}$ computes the master secret after receiving $T_S$ from $\pi_{j*}^{t^*}$, and then it sends $m_8 = T_C$. If the adversary decides to forward $m_8$ to $\pi_{j*}^{t^*}$, then the master secret of $\pi_{j*}^{t^*}$ is set equal to $R$. If $\pi_{j*}^{t^*}$ receives $T_{C'} \neq T_C$, then $\mathcal{A}_{\mathsf{prfodh}}$ queries its oracle to compute $ms' = \mathsf{PRF}(T_{C'}^v, label_1||r_C||r_S)$, and sets the master secret of $\pi_{j*}^{t^*}$ equal to $ms'$.

Note also that in any case algorithm $\mathcal{A}_{\mathsf{prfodh}}$ 'knows' the master secret of $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$, and thus is able to compute all further protocol messages (in particular the finished messages $fin_C$ and $fin_S$) and answer a potential $\mathsf{Reveal}$-query to $\pi_{j*}^{t^*}$ as required (note that there is no $\mathsf{Reveal}$-query to $\pi_{i*}^{s^*}$, as otherwise the experiment is aborted, due to Game 2). If $R = \mathsf{PRF}(g^{uv}, label_1||r_C||r_S)$, then the view of $\mathcal{A}$ is identical to Game 4, while if $R \xleftarrow{\$} \{0,1\}^\mu$ then it is identical to Game 5, which yields

$$\Pr\left[\mathsf{break}_4^{(1)}\right] \leq \Pr\left[\mathsf{break}_5^{(1)}\right] + \epsilon_{\mathsf{PRF\text{-}ODH}}$$

**Game 6** In this game, we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i*}^{s^*}$ with a random function. If $\pi_{j*}^{t^*}$ uses the same master secret $\widetilde{ms}$ as $\pi_{i*}^{s^*}$ (cf. Game 5), then the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{j*}^{t^*}$ is replaced as well. Of course the same random function is used for both oracles sharing the same $\widetilde{ms}$. In particular, this function is used to compute the `Finished` messages by both partner oracles.

Distinguishing Game 6 from Game 5 implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ breaking the security of the pseudo-random function $\mathsf{PRF}$, thus

$$\Pr\left[\mathsf{break}_5^{(1)}\right] \leq \Pr\left[\mathsf{break}_6^{(1)}\right] + \epsilon_{\mathsf{PRF}}$$

**Game 7** In Game 6, we have replaced the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ with a random function. Thus, the Server-`Finished` message expected by $\pi_{i*}^{s^*}$ is

$$fin_S^* = F_{\widetilde{ms}}(label_4 || \mathsf{H}(m_1 || \cdots || m_{10} || fin_C || m_{12}))$$

where $m_1 || \cdots || m_{10} || fin_C || m_{12}$ denotes the transcript of all messages sent and received by $\pi_{i*}^{s^*}$. In the next game, we would like to argue that the adversary is not able to predict $fin_S^*$, unless there is an oracle $\pi_{j*}^{t^*}$ having a matching conversation to $\pi_{i*}^{s^*}$, because $F_{\widetilde{ms}}$ is random. Before we can do so, we need to make sure that oracle $\pi_{j*}^{t^*}$ (the only other oracle potentially having access to $F_{\widetilde{ms}}$, due to Game 6) never evaluates $F_{\widetilde{ms}}$ on any input $label_4 || \mathsf{H}(m')$ with

$$m' \neq m_1 || \cdots || m_{10} || fin_C || m_{12} \quad \text{and} \quad \mathsf{H}(m') = \mathsf{H}(m_1 || \cdots || m_{10} || fin_C || m_{12}). \tag{3}$$

Therefore we add another abort condition. We abort the game, if oracle $\pi_{j*}^{t^*}$ ever evaluates the random function $F_{\widetilde{ms}}$ on an input $m'$ such that (3) holds. Since (3) implies that a collision for $\mathsf{H}$ is found, we can construct an adversary $\mathcal{A}_{\mathsf{H}}$ finding a hash collision in time $t \approx t'$ and with success probability $\epsilon_{\mathsf{H}}$, where

$$\Pr\left[\mathsf{break}_6^{(1)}\right] \leq \Pr\left[\mathsf{break}_7^{(1)}\right] + \epsilon_{\mathsf{H}}$$

**Game 8** Finally, we use that the *unique* (due to Game 7) hash of 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 that is only accessible to $\pi_{i*}^{s^*}$ and (possibly) $\pi_{j*}^{t^*}$ due to Game 6. This allows to show that any adversary has probability at most $2^{-\mu}$ of making oracle $\pi_{i*}^{s^*}$ accept without having a matching conversation to $\pi_{j*}^{t^*}$.

Thus, this game proceeds exactly like the previous game, except that the challenger now aborts if oracle $\pi_{i*}^{s^*}$ accepts without having a matching conversation to $\pi_{j*}^{t^*}$. Thus we have $\Pr\left[\mathsf{break}_8^{(1)}\right] = 0$.

The `Finished` messages are computed by evaluating a truly random function $F_{\widetilde{ms}}$. This function is only accessible to oracles sharing $\widetilde{ms}$, and evaluated on a unique hash value derived from the full transcript containing all previous messages. Thus, if there is no oracle having a matching conversation to $\pi_{i^*}^{s^*}$, the adversary receives no information about $F_{\widetilde{ms}}(label_4||\mathsf{H}(m_1||\cdots||m_{12}))$. Therefore we have $\Pr\left[\mathsf{break}_8^{(1)}\right] = 0$ and

$$\Pr\left[\mathsf{break}_7^{(1)}\right] \leq \Pr\left[\mathsf{break}_8^{(1)}\right] + \frac{1}{2^\mu} = \frac{1}{2^\mu}$$

Collecting probabilities from Game 0 to Game 8 yields Lemma 1. □

**Lemma 2.** *From any* Server*-adversary $\mathcal{A}$ that runs in time $t'$ with success probability $\epsilon_{\mathsf{server}}$, we can construct adversaries $\mathcal{A}_{\mathsf{sig}}$, $\mathcal{A}_{\mathsf{ddh}}$, $\mathcal{A}_{\mathsf{PRF}}$, and $\mathcal{A}_{\mathsf{H}}$ as in Theorem 1, with*

$$\epsilon_{\mathsf{server}} \leq d\ell\left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \frac{1}{2^\mu}\right)$$

where all quantities are defined as in Theorem 1.

*Proof.* Let $\mathsf{break}_\delta^{(2)}$ be the event that occurs when the first oracle accepts maliciously in the sense of Definition 9 with $\rho = \mathsf{Server}$ in Game $\delta$.

**Game 0** This game equals the AKE security experiment described in Sect. 4. Thus, for some $\epsilon_{\mathsf{server}}$ we have

$$\Pr\left[\mathsf{break}_0^{(2)}\right] = \epsilon_{\mathsf{server}}$$

**Game 1** In this game, we add an abort rule. The challenger aborts, if there exists any oracle $\pi_i^s$ that chooses a random nonce $r_C$ or $r_S$ which is not unique. With the same arguments as in Game 1 from the proof of Lemma 1, we have

$$\Pr\left[\mathsf{break}_0^{(2)}\right] \leq \Pr\left[\mathsf{break}_1^{(2)}\right] + \frac{(d\ell)^2}{2^\lambda}$$

**Game 2** This game is identical, except for the following. The challenger guesses two random indices $(i^*, s^*) \xleftarrow{\$} [\ell] \times [d]$. If there exists an oracle $\pi_i^s$ that 'accepts' maliciously, and $(i, s) \neq (i^*, s^*)$ and $\pi_i^s$ has $\rho \neq \mathsf{Server}$, then the challenger aborts the game. Note that if the first oracle $\pi_i^s$ that 'accepts' maliciously has $\rho = \mathsf{Server}$, then with probability $1/(d\ell)$ we have $(i, s) = (i^*, s^*)$, and thus

$$\Pr\left[\mathsf{break}_1^{(2)}\right] = d\ell \cdot \Pr\left[\mathsf{break}_2^{(2)}\right]$$

Note that in this game the adversary can only break the security of the protocol, if oracle $\pi_{i^*}^{s^*}$ is the first oracle that 'accepts' maliciously and has $\rho = \mathsf{Server}$, as otherwise the game is aborted.

**Game 3** The challenger proceeds as before, but we add an abort rule. We want to make sure that $\pi_{i*}^{s^*}$ receives as input exactly the Diffie–Hellman value $m_8 = T_C$ that was selected by some other uncorrupted oracle.

Technically, we abort and raise event $\mathsf{abort}_{\mathsf{sig}}$, if oracle $\pi_{i*}^{s^*}$ ever receives as input a message $m_7 = cert_C$ indicating intended partner $\Pi = j$ and message $m_9 = \sigma_C = \mathsf{SIG.Sign}(sk_C, m_1 || \ldots, ||m_8)$ such that $\sigma_C$ is a valid signature but there exists no oracle $\pi_j^t$ which has previously output $\sigma_C$. Clearly we have

$$\Pr\left[\mathsf{break}_2^{(2)}\right] \leq \Pr\left[\mathsf{break}_3^{(2)}\right] + \Pr[\mathsf{abort}_{\mathsf{sig}}]$$

Note that the experiment is aborted, if $\pi_{i*}^{s^*}$ does not accept *maliciously*, due to Game 2. This means that party $P_j$ must be $\tau_j$-corrupted with $\tau_j = \infty$ (i.e. not corrupted) when $\pi_{i*}^{s^*}$ accepts. To show that $\Pr[\mathsf{abort}_{\mathsf{sig}}] \leq \ell \cdot \epsilon_{\mathsf{sig}}$, we construct a signature forger $\mathcal{A}_{\mathsf{sig}}$ 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 3, except that it uses its chosen-message oracle to generate a signature under $pk_\phi$ when necessary.

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

$$\Pr\left[\mathsf{break}_2^{(2)}\right] \leq \Pr\left[\mathsf{break}_3^{(2)}\right] + \ell \cdot \epsilon_{\mathsf{sig}}$$

Note that in Game 3 oracle $\pi_{i*}^{s^*}$ receives as input a Diffie–Hellman value $T_C$ such that $T_C$ was chosen by another oracle, but not by the adversary. Note also that this oracle is unique, since the signature includes the client nonce $r_C$, which is unique due to Game 1. From now on, we denote this unique oracle with $\pi_{j*}^{t^*}$.

Note also that $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ share a premaster secret $pms = T_C^{t_S} = T_S^{t_C}$, where $T_C = g^{t_C}$ and $T_S = g^{t_S}$ for random exponents $t_S$ and $t_C$ chosen by $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$, respectively.

**Game 4** In this game, we replace the premaster secret $pms = g^{t_C t_S}$ shared by $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ with a random value $g^r$, $r \xleftarrow{\$} \mathbb{Z}_q$. The fact that the challenger has full control over the Diffie–Hellman shares $T_C$ and $T_S$ exchanged between $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$, 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 4 proceeds as before, but when $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ compute the premaster secret as $pms = g^{t_C t_S}$, the challenger replaces this value with a uniformly random value $\widetilde{pms} = g^r, r \xleftarrow{\$} \mathbb{Z}_p^*$, which is in the following used by both partner oracles. Suppose there exists an algorithm distinguishing Game 4 from Game 3. Then we can construct an algorithm $\mathcal{A}_{\mathsf{ddh}}$ breaking the DDH assumption as follows. Algorithm $\mathcal{A}_{\mathsf{ddh}}$ receives as input a DDH challenge $(g, g^u, g^v, g^w)$. The

challenger defines $T_C{:=}g^u$ and $T_S{:=}g^v$ for the Diffie–Hellman shares chosen by $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$, respectively. Instead of computing the Diffie–Hellman key as in Game 3, it sets $pms = g^w$ for both the 'client' and the 'server' oracle. Now if $w = uv$, then this game proceeds exactly like Game 3, while if $w$ is random than this game proceeds exactly like Game 4. $\mathcal{A}_{\mathsf{ddh}}$ runs in time $t \approx t'$ and $(t, \epsilon_{\mathsf{DDH}})$-breaks the DDH assumption, where $\epsilon_{\mathsf{DDH}}$ satisfies

$$\Pr\left[\mathsf{break}_3^{(2)}\right] \leq \Pr\left[\mathsf{break}_4^{(2)}\right] + \epsilon_{\mathsf{DDH}}$$

Note that in Game 4 the premaster secret of $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ is uniformly random, and independent of $T_C$ and $T_S$. This will provide us with the leverage to replace the function $\mathsf{PRF}(\widetilde{pms}, \cdot)$ with a truly random function in the next game.

**Game 5** In Game 5, we make use of the fact that the premaster secret $\widetilde{pms}$ of $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ is chosen uniformly random, and independent of $T_C$ and $T_S$. We thus replace the value $ms = \mathsf{PRF}(\widetilde{pms}, label_1||r_C||r_S)$ with a random value $\widetilde{ms}$.

Distinguishing Game 5 from Game 4 implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ $(t, \epsilon_{\mathsf{PRF}})$-breaking the security of the pseudo-random function $\mathsf{PRF}$ in time $t \approx t'$ with success probability $\epsilon_{\mathsf{PRF}}$, where

$$\Pr\left[\mathsf{break}_4^{(2)}\right] \leq \Pr\left[\mathsf{break}_5^{(2)}\right] + \epsilon_{\mathsf{PRF}}$$

**Game 6** In this game, we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ with a random function. Of course the same random function is used for both oracles $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$. In particular, this function is used to compute the `Finished` messages by both partner oracles.

Distinguishing Game 6 from Game 5 again implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ $(t, \epsilon_{\mathsf{PRF}})$-breaking the security of the pseudo-random function $\mathsf{PRF}$ in time $t \approx t'$ with success probability $\epsilon_{\mathsf{PRF}}$, where

$$\Pr\left[\mathsf{break}_5^{(2)}\right] \leq \Pr\left[\mathsf{break}_6^{(2)}\right] + \epsilon_{\mathsf{PRF}}$$

**Game 7** In Game 6, we have replaced the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ with a random function. Thus, the Client-`Finished` message expected by $\pi_{i*}^{s*}$ is

$$fin_C^* = F_{\widetilde{ms}}(label_3||\mathsf{H}(m_1||\cdots||m_{10})),$$

where $m_1||\cdots||m_{10}$ denotes the transcript of all messages sent and received by $\pi_{i*}^{s*}$. Again we would like to argue that the adversary is not able to predict $fin_C^*$, unless there is an oracle $\pi_{j*}^{t*}$ having a matching conversation to $\pi_{i*}^{s*}$, because $F_{\widetilde{ms}}$ is random.

Before we can do so, we need to make sure that oracle $\pi_{j*}^{t*}$ (the only other oracle potentially having access to $F_{\widetilde{ms}}$, due to Game 6) never evaluates $F_{\widetilde{ms}}$ on any input $label_3||\mathsf{H}(m')$ with

$$m' \neq m_1||\cdots||m_{10} \quad \text{and} \quad \mathsf{H}(m') = \mathsf{H}(m_1||\cdots||m_{10}). \tag{4}$$

Therefore we add another abort condition. We abort the game, if oracle $\pi_{j*}^{t*}$ ever evaluates the random function $F_{\widetilde{ms}}$ on an input $m'$ such that (4) holds. Since (4) implies that a collision for $\mathsf{H}$ is found, we can construct an adversary $\mathcal{A}_{\mathsf{H}}$ finding a hash collision in time $t \approx t'$ and with success probability $\epsilon_{\mathsf{H}}$, where

$$\Pr\left[\mathsf{break}_6^{(2)}\right] \leq \Pr\left[\mathsf{break}_7^{(2)}\right] + \epsilon_{\mathsf{H}}$$

**Game 8** Finally we use that the unique hash of the full transcript of all messages sent and received by $\pi_{i*}^{s*}$ is used to compute the `Finished` messages and that `Finished` messages are computed by evaluating a truly random function that is only accessible to $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ due to Game 7. This allows to show that any adversary has probability at most $\frac{1}{2^{\mu}}$ of making oracle $\pi_{i*}^{s*}$ accept without having a matching conversation to $\pi_{j*}^{t*}$.

Thus, this game proceeds exactly like the previous game, except that the challenger now aborts if oracle $\pi_{i*}^{s*}$ accepts without having a matching conversation to $\pi_{j*}^{t*}$. Therefore we have $\Pr\left[\mathsf{break}_8^{(1)}\right] = 0$.

The `Finished` messages are computed by evaluating a truly random function $F_{\widetilde{ms}}$, which is only accessible to oracles sharing $\widetilde{ms}$, and the full transcript containing all previous messages is used to compute the `Finished` messages. If there is no oracle having a matching conversation to $\pi_{i*}^{s*}$, the adversary receives no information about $F_{\widetilde{ms}}(label_3||m_1||\cdots||m_{10})$. Thus we have

$$\Pr\left[\mathsf{break}_7^{(2)}\right] \leq \Pr\left[\mathsf{break}_8^{(2)}\right] + \frac{1}{2^{\mu}} = \frac{1}{2^{\mu}}$$

Collecting probabilities from Game 0 to Game 8 yields Lemma 2. □

### 5.2. *Indistinguishability of Keys*

**Lemma 3.** *From any* $\mathsf{Test}$*-adversary* $\mathcal{A}$ *running in time* $t'$ *with success probability* $1/2 + \epsilon_{\mathsf{ke}}$*, we can construct adversaries* $\mathcal{A}_{\mathsf{ddh}}$ *and* $\mathcal{A}_{\mathsf{PRF}}$ *as in Theorem 1, with*

$$\epsilon_{\mathsf{ke}} \leq d\ell \cdot (\epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}})$$

*where all quantities are defined as in Theorem 1.*

*Proof.* Assume without loss of generality that $\mathcal{A}$ always asks a $\mathsf{Test}$-query such that all conditions in Property 2 of Definition 9 are satisfied. Let $\mathsf{break}_{\delta}^{(3)}$ denote the event that $b' = b$ in Game $\delta$, where $b$ is the random bit sampled by the $\mathsf{Test}$-query and $b'$ is either the bit output by $\mathcal{A}$ or (if $\mathcal{A}$ does not output a bit) chosen uniformly random by the challenger. Let $\mathsf{Adv}_{\delta} := \Pr\left[\mathsf{break}_{\delta}^{(3)}\right] - 1/2$ denote the *advantage* of $\mathcal{A}$ in Game $\delta$. Consider the following sequence of games.

**Game 0** This game equals the AKE security experiment described in Sect. 4. For some $\epsilon_{\mathsf{ke}}$, we have

$$\Pr\left[\mathsf{break}_0^{(3)}\right] = \frac{1}{2} + \epsilon_{\mathsf{ke}} = \frac{1}{2} + \mathsf{Adv}_0$$

Recall that we assume that $\mathcal{A}$ always asks a Test-query such that all conditions in Property 2 of Definition 9 are satisfied. In particular, it asks a Test-query to an oracle $\pi_i^s$ that 'accepts' after the $\tau_0$th query of $\mathcal{A}$ with intended partner $\Pi = j$, such that $P_j$ is $\tau_j$-corrupted with $\tau_j > \tau_0$. Since we also assume that $\mathcal{A}$ does *not* make any oracle accept maliciously (note this case is already considered in Lemmas 1 and 2), it will always hold that for any such oracle $\pi_i^s$ there exists a unique 'partner oracle' $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$, as the game is aborted otherwise.

**Game 1** The challenger in this game proceeds as before, but in addition guesses indices $(i^*, s^*) \xleftarrow{\$} [\ell] \times [d]$. It aborts and chooses $b'$ at random, if the adversary issues a $\mathsf{Test}(\pi_i^s)$-query with $(i, s) \neq (i^*, s^*)$. With probability $1/(d\ell)$, we have $(i, s) = (i^*, s^*)$, and thus

$$\mathsf{Adv}_0 \leq d\ell \cdot \mathsf{Adv}_1$$

Note that in Game 1 we know that $\mathcal{A}$ will issue a Test-query to oracle $\pi_{i^*}^{s^*}$. Note also that $\pi_{i^*}^{s^*}$ has a unique 'partner' due to Game 0. In the sequel, we denote with $\pi_{j^*}^{t^*}$ the unique oracle such that $\pi_{i^*}^{s^*}$ has a matching conversation to $\pi_{j^*}^{t^*}$, and say that $\pi_{j^*}^{t^*}$ is the *partner* of $\pi_{i^*}^{s^*}$.

**Game 2** Let $T_{i^*,s^*} = g^u$ denote the Diffie–Hellman share chosen by $\pi_{i^*}^{s^*}$, and let $T_{j^*,t^*} = g^v$ denote the share chosen by its partner $\pi_{j^*}^{t^*}$. Thus, both oracles compute the premaster secret as $pms = g^{uv}$.

The challenger in this game proceeds as before, but replaces the premaster secret $pms$ of $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ with a random group element $\widetilde{pms} = g^w$, $w \xleftarrow{\$} \mathbb{Z}_q$. Note that both $g^u$ and $g^v$ are chosen by oracles $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$, respectively, as otherwise $\pi_{i^*}^{s^*}$ would not have a matching conversation to $\pi_{j^*}^{t^*}$ and the game would be aborted.

Suppose that there exists an algorithm $\mathcal{A}$ distinguishing Game 2 from Game 1. Then we can construct an algorithm $\mathcal{A}_{\mathsf{ddh}}$ breaking the DDH assumption as follows. $\mathcal{A}_{\mathsf{ddh}}$ receives as input $(g, g^u, g^v, g^w)$. It implements the challenger for $\mathcal{A}$ as in Game 1, except that it sets $T_{i^*,s^*} := g^u$ and $T_{j^*,t^*} := g^v$, and the premaster secret of $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ equal to $pms := g^w$. Note that $\mathcal{A}_{\mathsf{ddh}}$ can simulate all messages exchanged between $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ properly, in particular the finished messages using knowledge of $pms = g^w$. Since all other oracles are not modified, $\mathcal{A}_{\mathsf{ddh}}$ can simulate these oracles properly as well.

If $w = uv$, then the view of $\mathcal{A}$ when interacting with $\mathcal{A}_{\mathsf{ddh}}$ is identical to Game 1, while if $w \xleftarrow{\$} \mathbb{Z}_q$, then it is identical to Game 2. Thus, the DDH assumption implies that

$$\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{DDH}}$$

**Game 3** In Game 3, we make use of the fact that the premaster secret $\widetilde{pms}$ of $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ is chosen uniformly random. We thus replace the value $ms = \mathsf{PRF}(\widetilde{pms}, label_1 ||r_C||r_S)$ with a random value $\widetilde{ms}$.

Distinguishing Game 3 from Game 2 implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ breaking the security of the pseudo-random function PRF, thus

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{PRF}}$$

**Game 4** In this game, we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$ with a random function $F_{\widetilde{ms}}$. Of course the same random function is used for both oracles $\pi_{i*}^{s*}$ and $\pi_{j*}^{t*}$. In particular, this function is used to compute the key material as

$$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C} := F_{\widetilde{ms}}(label_2||r_C||r_S)$$

Distinguishing Game 4 from Game 3 again implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ breaking the security of the pseudo-random function PRF. Moreover, in Game 4 the adversary always receives a random key in response to a Test-query and thus receives no information about $b'$, which implies $\mathsf{Adv}_4 = 0$ and

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{PRF}} = \epsilon_{\mathsf{PRF}}.$$

Collecting probabilities from Game 0 to Game 4 yields Lemma 3 □

## 6. ACCE Protocols

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

**Pre-accept phase** In this phase, a 'handshake protocol' is executed. In terms of functionality, this protocol is an AKE protocol as in Sect. 4, 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 9). This phase ends, when both communication partners reach an `accept`-state (i.e. $\Lambda =$ '`accept`').

**Post-accept phase** This phase is entered, when both communication partners reach an `accept`-state. In this phase, data can be transmitted, encrypted, and authenticated with key $k$.

The prime example for an ACCE protocol is TLS. Here, the pre-accept phase consists of the TLS Handshake protocol. In the post-accept phase, encrypted and authenticated data are transmitted over the TLS Record Layer.

To define security of ACCE protocols, we combine the security model for authenticated key exchange from Sect. 4 with stateful length-hiding encryption in the sense

of [81]. 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 from Sect. 4, except for a few simple modifications. We extend the model such that in the post-accept phase an adversary is also able to 'inject' chosen-plaintexts by making an Encrypt-query,[11] and chosen-ciphertexts by making a Decrypt-query. Each oracle $\pi_i^s$ keeps as additional internal state a bit $b_i^s \xleftarrow{\$} \{0,1\}$, chosen at random at the beginning of the game.

Moreover, for the post-accept phase each oracle $\pi_i^s$ keeps additional variables $(u_i^s, v_i^s, C_i^s, \theta_i^s)$. Variables $u_i^s, v_i^s$ are counters, which are initialized to $(u_i^s, v_i^s) := (0,0)$. To simplify our notation in the sequel, we additionally define $u_0^0 := 0$. Variable $C_i^s$ is a list of ciphertexts, which initially is empty. We write $C_i^s[u]$ to denote the $u$-th entry of $C_i^s$. Variable $\theta_i^s$ stores a pair of indices $\theta_i^s \in [\ell] \cup \{0\} \times [d] \cup \{0\}$. If oracle $\pi_i^s$ accepts having a matching conversation to some other oracle $\pi_j^t$, then $\theta_i^s$ is set to $\theta_i^s := (j,t)$.[12] Otherwise it is set to $\theta_i^s := (0,0)$.

In the sequel, we will furthermore assume that the key $k$ consists of two different keys $k = (k_{\mathsf{enc}}^{\rho}, k_{\mathsf{dec}}^{\rho})$ for encryption and decryption. Their order depends on the role $\rho \in \{\mathsf{Client}, \mathsf{Server}\}$ of oracle $\pi_i^s$. This is the case for TLS (see Sect. 3).

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 AKE model from Sect. 4, except that it replies with an error symbol $\perp$ if oracle $\pi_i^s$ has state $\Lambda =$ '`accept`'. (Send-queries in an accept-state are handled by the Decrypt-query below).
- $\mathsf{Reveal}(\pi_i^s)$ and $\mathsf{Corrupt}(P_i)$: These queries are identical to the corresponding queries in the AKE model from Sect. 4.
- $\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \mathsf{len}, H)$: This query takes as input two messages $m_0$ and $m_1$, length parameter len, and header data $H$. If $\Lambda \neq$ '`accept`', then $\pi_i^s$ returns $\perp$.

  Otherwise, it proceeds as depicted in Fig. 4, depending on the random bit $b_i^s \xleftarrow{\$} \{0,1\}$ sampled by $\pi_i^s$ at the beginning of the game and the internal state variables of $\pi_i^s$.
- $\mathsf{Decrypt}(\pi_i^s, C, H)$: This query takes as input a ciphertext $C$ and header data $H$. If $\pi_i^s$ has $\Lambda \neq$ '`accept`' then $\pi_i^s$ returns $\perp$. Otherwise, it proceeds as depicted in Fig. 4.

*Remark 7.* Note that in the case of TLS, a message encrypted by some oracle $\pi_i^s$ can only be decrypted by its 'partner' oracle, as different keys are used for the different communication directions (i.e. a single oracle uses different keys for encryption and decryption).

[11] This models that an adversary may trick one party into sending some adversarially chosen data. A practical example for this attack scenario is cross-site request forgeries [90] on web servers, or Bard's chosen-plaintext attacks on SSL3.0 [5,6].

[12] If there is more than one such oracle, the first in lexicographical order is chosen.

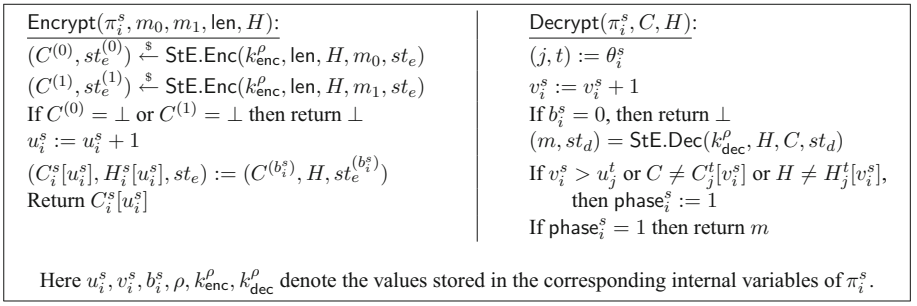


**Fig. 4.** Encrypt and Decrypt oracles in the ACCE security experiment.

### 6.2. *Security Definition*

Security of ACCE protocols is defined by requiring that (i) the protocol is a secure authentication protocol and (ii) in the post-accept phase all data are transmitted over an authenticated and confidential channel in the sense of Definition 6.

Again this notion is captured by 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 $\ell$ 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. Now the adversary may start issuing $\mathsf{Send}^{\mathsf{pre}}$, Reveal, Corrupt, Encrypt, and Decrypt queries. Finally, the adversary outputs a triple $(i, s, b')$ and terminates.

**Definition 10.** Assume a 'benign' adversary $\mathcal{A}$, which picks two arbitrary oracles $\pi_i^s$ and $\pi_j^t$ and performs a sequence of $\mathsf{Send}^{\mathsf{pre}}$-queries by faithfully forwarding all messages of the pre-accept phase between $\pi_i^s$ and $\pi_j^t$. Let $k_i^s = (k^{\mathsf{Client}}_{\mathsf{enc}}, k^{\mathsf{Client}}_{\mathsf{dec}})$ denote the key computed by $\pi_i^s$ and let $k_j^t = (k^{\mathsf{Server}}_{\mathsf{dec}}, k^{\mathsf{Server}}_{\mathsf{enc}})$ denote the key computed by $\pi_j^t$. We say that an ACCE protocol is *correct*, if for this benign adversary and any two oracles $\pi_i^s$ and $\pi_j^t$ always holds that

1. both oracles have $\Lambda = \texttt{accept}$,
2. $k_i^s = k_j^t \in \mathcal{K}$.

Furthermore we require that for all messages $m \in \{0,1\}^*$, lengths fields $\mathsf{len} \in \mathbb{N}$ with $\mathsf{len} \geq |m|$, roles $\rho \in \{\mathsf{Client}, \mathsf{Server}\}$, keys $k = (k^\rho_{\mathsf{enc}}, k^\rho_{\mathsf{dec}})$, headers $H \in \{0,1\}^*$, and encryption/decryption states $st_e, st_d \in \{0,1\}^*$ holds that $\mathsf{StE.Dec}(k^\rho_{\mathsf{dec}}, H, \mathsf{StE.Enc}(k^\rho_{\mathsf{enc}}, \mathsf{len}, H, m, st_e), st_d) = m$.

**Definition 11.** We say that an adversary $(t, \epsilon)$-*breaks* an ACCE protocol, if $\mathcal{A}$ runs in time $t$, and at least one of the following two conditions holds:

1. When $\mathcal{A}$ terminates, then with probability at least $\epsilon$ there exists an oracle $\pi_i^s$ such that
   - $\pi_i^s$ 'accepts' when $\mathcal{A}$ issues its $\tau_0$th query with partner $\Pi = j$, and
   - $P_j$ is $\tau_j$-corrupted with $\tau_0 < \tau_j$, and

- $\mathcal{A}$ did not issue a Reveal-query to $\pi_i^s$, or to an oracle $\pi_j^t$, such that $\pi_j^t$ accepted while having a matching conversation to $\pi_i^s$ (if such an oracle exists) and
- there is no unique oracle $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$.

If an oracle $\pi_i^s$ accept in the above sense, then we say that $\pi_i^s$ accepts *maliciously*.

2. When $\mathcal{A}$ terminates and outputs a triple $(i, s, b')$ such that
   - $\pi_i^s$ 'accepts' when $\mathcal{A}$ issues its $\tau_0$th query with intended partner $\Pi = j$, and
   - $P_j$ is $\tau_j$-corrupted with $\tau_0 < \tau_j$, and
   - $\mathcal{A}$ did not issue a Reveal-query to $\pi_i^s$, nor to $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$ (if such an oracle exists),

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

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

   If an adversary $\mathcal{A}$ outputs $(i, s, b')$ such that $b' = b_i^s$ and the above conditions are met, then we say that $\mathcal{A}$ *answers the encryption-challenge correctly*.

We say that an ACCE protocol is $(t, \epsilon)$-*secure*, if it is correct and there exists no adversary that $(t, \epsilon)$-*breaks* it.

*Remark 8.* In comparison with the AKE Definition (Definition 9), we included an additional condition in Definition 11. Namely, we require that $\mathcal{A}$ did not issue a Reveal-query to an oracle $\pi_j^t$, such that $\pi_j^t$ accepted while having a matching conversation to $\pi_i^s$, to prevent a trivial attack, which is possible for all ACCE protocols in which the last message (of the pre-accept phase) $m_x$ is computed by some probabilistic algorithm $m_x \xleftarrow{\$} \mathsf{F}(k, in, r)$, where $k$ is the session key $k$, $in$ is some publicly known input (like for instance the transcript of all messages), and $r$ is the randomness of the algorithm. For simplicity, we assume that some server-oracle $\pi_j^t$ sends the last message to a client-oracle $\pi_i^s$ (its intended partner). The adversary can now proceed as follows:

By definition, $\pi_j^t$ has to accept after sending $m_x$ (without knowing if this message was faithfully received by $\pi_i^s$). Without the additional restriction, it would be possible to ask a Reveal-query to $\pi_j^t$ and thus to learn the session key $k$. Note that an active adversary at this point may have dropped the message $m_x$. $\mathcal{A}$ can now use the key $k$, compute a similar message $m'_x$ by evaluating the same function with fresh randomness $r'$ as $m'_x \xleftarrow{\$} \mathsf{F}(k, in, r')$ over the same input $in$ that $\pi_j^t$ used and send $m'_x \neq m_x$ to $\pi_i^s$. If $m_x \neq m'_x$, then $\pi_i^s$ will accept without having a matching conversation to $\pi_j^t$. This issue was pointed out to us by [68] and observed independently in [28].

*Remark 9.* Also note that the above definition even allows to corrupt the oracle $\pi_i^s$ whose internal secret bit the adversary tries to determine (Property 2). Thus, protocols secure with respect to this definition provide *perfect forward secrecy*. Similar to the AKE definitions, we again allow the 'accepting' oracle to be corrupted even *before* it reaches an `accept`-state, which provides security against *key-compromise impersonation* attacks (Property 1).

Also note that by explicitly differentiating between authentication and confidentiality, we are able to model KCI attacks against protocols without perfect forward secrecy.

### 6.3. *Relation to the AKE Security Definition from Sect. 4*

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. for the adversary indistinguishable from random) 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 security in the sense of Definition 9, and the 'symmetric part' consists of a suitable authenticated symmetric encryption scheme (e.g. secure according to Definition 6).

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 Definition 9, and thus easier to achieve (possibly from weaker hardness assumptions, or by more efficient protocols).

We stress that our ACCE definition is mainly motivated by the fact that security models based on key indistinguishability do not allow for a security analysis of full TLS, as detailed in "Introduction". We do not want to propose ACCE as a new security notion for key exchange protocols, since it is very complex and the modular two-step approach seems more useful in general.

### 6.4. *Relation to the Notion of Secure Network Channels*

Our notion of authenticated and confidential channel establishment protocols is related to but different from the notion of secure network channel protocols provided by Canetti and Krawczyk [35]. Roughly, they define a network channel protocol to be a combination of a secure (with respect to key indistinguishability) key exchange protocol and a symmetric authentication or encryption scheme that is used for message exchange. At the same time they strictly separate the key exchange phase from the message exchange phase by requiring that the keys produced in the key exchange protocol expire in the key exchange protocol before they are used in the symmetric primitive. We stress that this, as detailed before, does not allow to prove the security of protocols like TLS where the two protocol phases overlap. The adversary's capabilities to access the encryption and decryption algorithms are similar to ours. A noteworthy difference is that they deal with replay attacks rather informally by requiring that each message contains a unique message identifier that can be checked by the receiver. Through the incorporation of the stateful LHAE definition into our model, we cover replay attacks explicitly.

## 7. TLS with Ephemeral Diffie–Hellman Is a Secure ACCE Protocol

Again we will consider three types of adversaries:

1. Client-adversaries that succeed in making a client-oracle accept maliciously.
2. Server-adversaries that succeed in making a server-oracle accept maliciously.
3. Adversaries that do not make any oracle accept maliciously. We call such an adversary an encryption-adversary.

**Theorem 2.** *From any adversary that $(t', \epsilon_{\mathsf{tls}})$-breaks the truncated ephemeral Diffie–Hellman TLS Handshake protocol in the sense of Definition 11, we can construct an adversary $\mathcal{A}_{\mathsf{PRF}}$ that $(t, \epsilon_{\mathsf{PRF}})$-breaks the security of* PRF*, $\mathcal{A}_{\mathsf{sig}}$ that $(t, \epsilon_{\mathsf{sig}})$-breaks the security of the signature scheme, $\mathcal{A}_{\mathsf{ddh}}$ that $(t, \epsilon_{\mathsf{DDH}})$-breaks the DDH assumption in the group $G$ used to compute the TLS-DHE premaster secret, $\mathcal{A}_{\mathsf{H}}$ that $(t, \epsilon_{\mathsf{H}})$-breaks the collision resistance of* H*, $\mathcal{A}_{\mathsf{prfodh}}$ that $(t, \epsilon_{\mathsf{prfodh}})$-breaks the* PRF-ODH*-problem with respect to $G$ and* PRF*, and $\mathcal{A}_{\mathsf{sLHAE}}$ that $(t, \epsilon_{\mathsf{sLHAE}})$-breaks the stateful symmetric encryption scheme, with $t \approx t'$ and the following lower bounds on the success probabilities of the constructed adversaries.*

- *If $\mathcal{A}$ is a* Client*-adversary, then it holds that*

$$\epsilon_{\mathsf{tls}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + d\ell \left( \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{slhae}} + \frac{1}{2^{\mu}} \right) \right)$$

- *If $\mathcal{A}$ is a* Server*-adversary, then it holds that*

$$\epsilon_{\mathsf{tls}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{slhae}} + \frac{1}{2^{\mu}} \right)$$

- *If $\mathcal{A}$ is an encryption-adversary, then it holds that*

$$\epsilon_{\mathsf{tls}} \leq d\ell \left( \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{sLHAE}} \right)$$

*Recall here that $\ell$ denotes the number of parties in the security model, $d$ the number of sessions per party, $\mu$ the output length of* PRF*, and $\lambda$ the length of the nonces $r_C$ and $r_S$.*

We prove Theorem 2 by the following three lemmas.

**Lemma 4.** *From any* Client*-adversary $\mathcal{A}$ that runs in time $t'$ with success probability $\epsilon_{\mathsf{client}}$, we can construct adversaries $\mathcal{A}_{\mathsf{sig}}$, $\mathcal{A}_{\mathsf{ddh}}$, $\mathcal{A}_{\mathsf{PRF}}$, $\mathcal{A}_{\mathsf{H}}$, and $\mathcal{A}_{\mathsf{sLHAE}}$ as in Theorem 2, with*

$$\epsilon_{\mathsf{client}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + d\ell \left( \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{slhae}} + \frac{1}{2^{\mu}} \right) \right)$$

*where all quantities are defined as in Theorem 2.*

**Lemma 5.** *From any* Server*-adversary* $\mathcal{A}$ *that runs in time* $t'$ *with success probability* $\epsilon_{\mathsf{server}}$*, we can construct adversaries* $\mathcal{A}_{\mathsf{sig}}$*,* $\mathcal{A}_{\mathsf{ddh}}$*,* $\mathcal{A}_{\mathsf{PRF}}$*,* $\mathcal{A}_{\mathsf{H}}$*, and* $\mathcal{A}_{\mathsf{sLHAE}}$ *as in Theorem 2, with*

$$\epsilon_{\mathsf{server}} \leq d\ell \left( \frac{d\ell}{2^{\lambda}} + \ell \cdot \epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{slhae}} + \frac{1}{2^{\mu}} \right)$$

*where all quantities are defined as in Theorem 2.*

The bounds on $\epsilon_{\mathsf{client}}$ and $\epsilon_{\mathsf{server}}$ in Lemma 4 and Lemma 5 are derived almost exactly as in the proofs of Lemma 1 and Lemma 2. We only extend both proofs by one game-hop that exploits the sLHAE security of the encryption scheme. This is necessary, as an adversary can violate the matching conversations definition and thus make an oracle maliciously accept, by creating a new, valid encryption of $fin_C$ (or $fin_S$), which is distinct from the ciphertext output by the corresponding client (or server) oracle.

*Remark 10.* Note that although in the non-truncated version of TLS the Client Finished message $fin_C$ is sent encrypted, the Server Finished message is always computed over the plaintext handshake messages. Thus, the Server Finished message computed in the unmodified version of TLS equals the Finished message that is computed in the truncated TLS version (assuming that the same parameters, randomness, etc. are used). In order to follow the same reasoning in the ACCE setting, we only need to make sure that the adversary is not able to produce *new* valid symmetric encryptions of finished messages. This is provided by the sLHAE security of the symmetric encryption scheme.

**Lemma 6.** *From any encryption-adversary* $\mathcal{A}$ *running in time* $t'$ *with success probability* $1/2 + \epsilon_{\mathsf{enc}}$*, we can construct adversaries* $\mathcal{A}_{\mathsf{ddh}}$*,* $\mathcal{A}_{\mathsf{PRF}}$*, and* $\mathcal{A}_{\mathsf{sLHAE}}$ *as in Theorem 2, with*

$$\epsilon_{\mathsf{enc}} \leq d\ell \left( \epsilon_{\mathsf{DDH}} + 2 \cdot \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{sLHAE}} \right)$$

*where all quantities are defined as in Theorem 2.*

The proof of this lemma again extends the proof of Lemma 3 by one game-hop that exploits the sLHAE security of the encryption scheme.

*Proof.* Assume without loss of generality that $\mathcal{A}$ always outputs $(i, s, b')$ such that all conditions in Property 2 of Definition 11 are satisfied. Let $\mathsf{break}_{\delta}^{(4)}$ denote the event that $b' = b_i^s$ in Game $\delta$, where $b_i^s$ is the random bit sampled by $\pi_i^s$, and $b'$ is either the bit output by $\mathcal{A}$ or (if $\mathcal{A}$ does not output a bit) chosen uniformly random by the challenger. Let $\mathsf{Adv}_{\delta} := \Pr[\mathsf{break}_{\delta}^{(4)}] - 1/2$ denote the *advantage* of $\mathcal{A}$ in Game $\delta$. Consider the following sequence of games.

**Game 0** This game equals the ACCE security experiment described in Sect. 6. For some $\epsilon_{\mathsf{enc}}$, we have

$$\Pr\left[\mathsf{break}_0^{(3)}\right] = \frac{1}{2} + \epsilon_{\mathsf{enc}} = \frac{1}{2} + \mathsf{Adv}_0$$

Recall that we consider an encryption-adversary, which does not make any oracle accept maliciously and that we assume that $\mathcal{A}$ always outputs $(i, s, b')$ such that all conditions in Property 2 of Definition 11 are satisfied. In particular, it outputs $(i, s, b')$ such that $\pi_i^s$ 'accepts' after the $\tau_0$th query of $\mathcal{A}$ with intended partner $\Pi = j$, and $P_j$ is $\tau_j$-corrupted with $\tau_j > \tau_0$. Note that in Game 0 for any such oracle $\pi_i^s$ there exists a unique 'partner oracle' $\pi_j^t$ such that $\pi_i^s$ has a matching conversation to $\pi_j^t$, as the game is aborted otherwise.

**Game 1** The challenger in this game proceeds as before, but in addition guesses indices $(i^*, s^*) \xleftarrow{\$} [\ell] \times [d]$. It aborts and chooses $b'$ at random, if the adversary outputs $(i, s, b')$ with $(i, s) \neq (i^*, s^*)$. With probability $1/(d\ell)$ we have $(i, s) = (i^*, s^*)$, and thus

$$\mathsf{Adv}_0 \leq d\ell \cdot \mathsf{Adv}_1$$

Note that in Game 1 we know that $\mathcal{A}$ will output $(i^*, s^*, b')$. Note also that $\pi_{i^*}^{s^*}$ has a unique 'partner' due to Game 0. In the sequel, we denote with $\pi_{j^*}^{t^*}$ the unique oracle such that $\pi_{i^*}^{s^*}$ has a matching conversation to $\pi_{j^*}^{t^*}$, and say that $\pi_{j^*}^{t^*}$ is the *partner* of $\pi_{i^*}^{s^*}$.

**Game 2** The challenger in this game proceeds as before, but replaces the premaster secret $pms$ of $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ with a random group element $\widetilde{pms} = g^w$, $w \xleftarrow{\$} \mathbb{Z}_q$. Note that both $g^u$ and $g^v$ are chosen by oracles $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$, respectively, as otherwise $\pi_{i^*}^{s^*}$ would not have a matching conversation to $\pi_{j^*}^{t^*}$ and the game would be aborted. With the same arguments as in Game 2 in the proof of Lemma 3, we can construct an adversary $\mathcal{A}_{\mathsf{ddh}}$ which runs in time $t \approx t'$ and such that

$$\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{DDH}}$$

**Game 3** As in Game 3 in the proof of Lemma 3, we now make use of the fact that the premaster secret $\widetilde{pms}$ of $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ is chosen uniformly random. We thus replace the value $ms = \mathsf{PRF}(\widetilde{pms}, label_1||r_C||r_S)$ with a random value $\widetilde{ms}$.

Distinguishing Game 3 from Game 2 implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ breaking the security of the pseudo-random function $\mathsf{PRF}$; thus

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{PRF}}$$

**Game 4** As in Game 3 in the proof of Lemma 3, we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ with a random function $F_{\widetilde{ms}}$. Of course the same random function is used for both oracles $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$. In particular, this function is used to compute the key material as

$$K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C} := F_{\widetilde{ms}}(label_2||r_C||r_S)$$

Distinguishing Game 4 from Game 3 again implies an algorithm $\mathcal{A}_{\mathsf{PRF}}$ breaking the security of the pseudo-random function $\mathsf{PRF}$, thus we have

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{PRF}}$$

Note that in Game 4 the key material $K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C}$ of oracles $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ is uniformly random and independent of all TLS Handshake messages exchanged in the pre-accept phase.

**Game 5** Now we use that the key material $K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{C\to S}||K_{\mathsf{mac}}^{S\to C}$ used by $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ in the stateful symmetric encryption scheme uniformly at random and independent of all TLS Handshake messages.

In this game, we construct a simulator $\mathcal{A}_{\mathsf{sLHAE}}$ that uses a successful ACCE adversary $\mathcal{A}$ to break the security of the underlying sLHAE-secure symmetric encryption scheme (Definition 6). By assumption, the simulator $\mathcal{A}_{\mathsf{sLHAE}}$ is given access to an encryption oracle $\mathsf{Encrypt}$ and a decryption oracle $\mathsf{Decrypt}$. $\mathcal{A}_{\mathsf{sLHAE}}$ embeds the sLHAE experiment by simply forwarding all $\mathsf{Encrypt}(\pi_{i*}^{s^*},\cdot)$ queries to $\mathsf{Encrypt}$, and all $\mathsf{Decrypt}(\pi_{j*}^{t^*},\cdot)$ queries to $\mathsf{Decrypt}$. Otherwise it proceeds as the challenger in Game 4.

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

$$\mathsf{Adv}_4 = \mathsf{Adv}_5$$

If $\mathcal{A}$ outputs a triple $(i^*, s^*, b')$, then $\mathcal{A}_{\mathsf{sLHAE}}$ forwards $b'$ to the sLHAE experiment. Otherwise it outputs a random bit. Since the simulator essentially relays all messages it is easy to see that an adversary $\mathcal{A}$ having advantage $\epsilon'$ yields an adversary $\mathcal{A}_{\mathsf{sLHAE}}$ against the sLHAE security of the encryption scheme with success probability at least $1/2 + \epsilon'$.

Since by assumption any adversary has advantage at most $\epsilon_{\mathsf{sLHAE}}$ in breaking the sLHAE security of the symmetric encryption scheme, we have

$$\mathsf{Adv}_5 \leq 1/2 + \epsilon_{\mathsf{sLHAE}}$$

□

## 8. On Proving Security of TLS-DHE from Standard Assumptions

In this section, we illustrate why we had to make the $\mathsf{PRF\text{-}ODH}$ assumption in the proofs of Lemmas 1 and 4 and why it is possible to prove Lemmas 2 and 5 based on the standard DDH assumption. In order to allow a comprehensive exposition, let us consider the simplified protocol described in Fig. 5 as an abstraction of the TLS-DHE Handshake.

Suppose we are given an adversary which always makes $\mathsf{Client}$-oracle $C{:=}\pi_i^s$ (i.e. a particular oracle $\pi_i^s$ with $\rho = \mathsf{Client}$) accept maliciously with intended partner $\Pi = S$. We will call this type of adversary a $\mathsf{Client}$-*adversary*. Suppose we want to argue that the adversary is not able to forge the $fin_S$-message received by $C$ (which we would have to, since the $fin_S$-message is the only message that cryptographically protects all messages previously received by $\pi_i^s$, and thus is required to ensure that $\pi_i^s$ has a matching conversation), and that we want to assume only that the $\mathsf{PRF}$ is secure in the standard sense (Definition 3). Then at some point in the proof we would have to replace the premaster secret computed by $\pi_i^s$ as $pms = T_S^{tC} = g^{tC\,tS}$ with an independent random value.

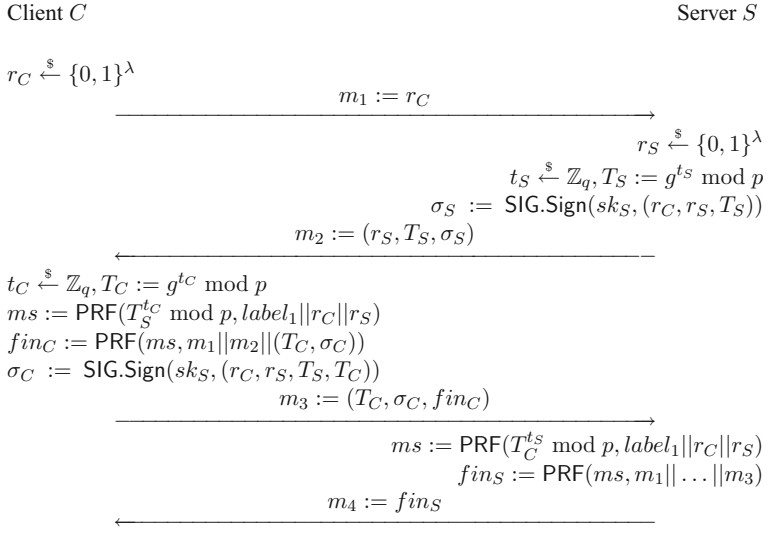


**Fig. 5.** Abstraction of the TLS-DHE Handshake.

Note that in order to replace $pms$ with a random value and argue in the proof with indistinguishability, we must not know any of the exponents $t_C$ and $t_S$ in $T_C = g^{t_C}$ and $T_S = g^{t_S}$, as otherwise we can trivially distinguish the real $pms = g^{t_C t_S}$ from a random $pms'$. The problematic property of TLS-DHE is that an adversary may test whether the challenger 'knows' $t_S$, and then make $\mathsf{Client}$-oracle $\pi_i^s$ accept maliciously only if this holds. This works as follows.

1. The adversary establishes a communication between two oracles $\pi_i^s$ (representing the client $C$) and $\pi_j^t$ (representing the server $S$) by simply forwarding the messages $m_1$ and $m_2$ between $C$ and $S$.
2. $C$ will respond with $m_3 = (T_C, \sigma_C, fin_C)$. This message is not forwarded.
3. Instead, the adversary corrupts some party $P^* \notin \{P_i, P_j\}$ and obtains the secret key $sk^*$ of this party. Then it computes

   (a) $T^* := g^{t^*} \bmod p$ for random $t^* \xleftarrow{\$} \mathbb{Z}_q$,
   (b) $\sigma^* := \mathsf{SIG.Sign}(sk^*; (r_C, r_S, T_S, T^*))$ using the corrupted key $sk^*$,
   (c) $ms^* := \mathsf{PRF}(T_S^{t^*}, label_1 || r_C || r_S)$ using knowledge of $t^*$, and
   (d) $fin_C^* := \mathsf{PRF}(ms^*, m_1 || m_2 || (T^*, \sigma^*))$.

   and sends $m_3^* := (T^*, \sigma^*, fin_C^*)$ to $S$. Note that $S$ cannot determine that its communication partner has changed, because any messages previously received by $S$ were perfectly anonymous.
4. If $S$ responds with a correct $fin_S^*$ message (note that the adversary is able to compute all keys, in particular $pms^* := T_S^{t^*}$, since it 'knows' $t^*$, and thus is able to verify the validity of $fin_S^*$), then adversary concludes that the challenger 'knows' $t_S$ and forges the required $fin_S$-message (e.g. by breaking CDH) to make $\pi_i^s$ accept without matching conversations. Otherwise the adversary aborts.

Note that the above adversary is a valid, successful adversary in the real security experiment. It does not issue any Reveal-query and only one Corrupt-query to an unrelated party, such that the intended communication partner $\Pi = S$ of $C = \pi_i^s$ remains uncorrupted, but still it makes $C = \pi_i^s$ 'accept' and there is no oracle that $C$ has a matching conversation to.

However, we will not be able to use this adversary in a simulated security experiment where the challenger does not know the exponent $t_S$ of $T_S = g^{t_S}$. Intuitively, the reason is that in this case the challenger would *first* have to compute the `Finished`-message $fin_S^*$, where

$$fin_S^* = \mathsf{PRF}(ms, m_1||\ldots||m_3) \quad \text{and} \quad ms = \mathsf{PRF}(T_S^{t^*}, label_1||r_C||r_S),$$

but 'knowing' neither $t_S = \log T_S$, nor $t^* = \log T^*$. This is the technical problem we are faced with, if we want to prove security under a standard assumption like DDH. Under the PRF-ODH assumption, we can, however, use the given oracle to compute first $ms$, and from this the `Finished`-message $fin_S^*$.

Server-adversaries Interestingly, the above technical problem does not appear if we consider Server-adversaries (i.e. adversaries that make an oracle $\pi_i^s$ accept maliciously with $\rho = \mathsf{Server}$) instead of Client-adversaries. This is due to the asymmetry of the TLS-DHE Handshake protocol. The reason is that in this case the adversary is not allowed to corrupt the intended partner of the server (in order to exclude trivial attacks), and is therefore not able to inject an adversarially chosen Diffie–Hellman share $T^*$. Note here that the signature sent from the client to the server is computed over *both* Diffie–Hellman shares received and chosen by the client. Therefore in this case the server is able to verify whether its intended partner has received the correct Diffie–Hellman share, and thus the standard DDH assumption is sufficient to prove Lemma 2.

Disallowing corruptions One possibility to circumvent the above problem, and thus to avoid the PRF-ODH assumption, is to consider a weaker security model. If we disallow Corrupt-queries in the model, then the adversary will not be able to inject an adversarially chosen, validly signed Diffie–Hellman share. This prevents the above 'test' and again allows a proof under the DDH assumption. However, a security model without corruptions is rather weak. Albeit it may be reasonable for certain applications, it is certainly not adequate for the way how TLS-DHE is used on the Internet.

Adopting $\Sigma_0$ to TLS. In [36] Canetti and Krawczyk describe a protocol called $\Sigma_0$, which exhibits many similarities to the TLS-DHE Handshake protocol and the simplified protocol from Fig. 5, but is provably secure under standard assumptions (in particular under DDH instead of PRF-ODH). Let us discuss why the differences between $\Sigma_0$ and TLS-DHE, albeit subtle, are crucial.

In Fig. 6, we describe a simple variant $\Sigma$ of $\Sigma_0$, which essentially extends $\Sigma_0$ with a server nonce $r_S$ and replaces the MAC computed over identities in $\Sigma_0$ with a MAC (implemented with PRF) computed over all previous protocol messages. Note that these messages include the identities, so the security analysis of $\Sigma_0$ carries over to $\Sigma$.

The major difference between $\Sigma_0$ and TLS-DHE is that the client 'accepts' already after receiving $m_2$. There is no message $m_4$ sent from the server to the client (which

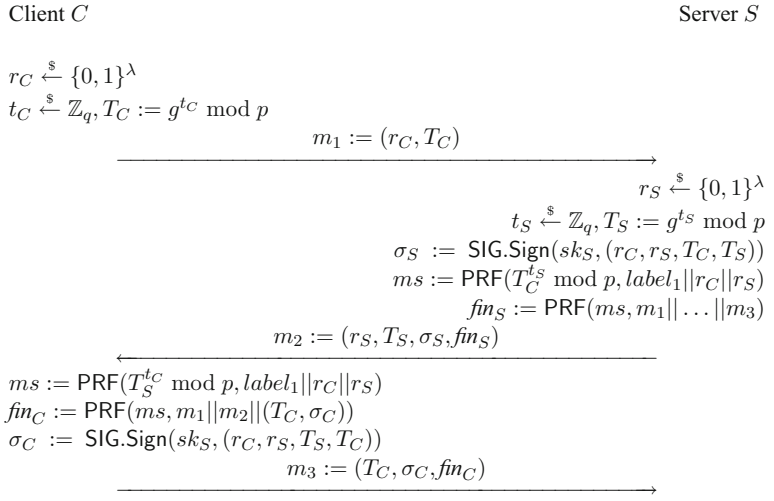


**Fig. 6.** Provably secure TLS-DHE adopting $\Sigma_0$ variant, adopted to the client/server setting and our notation. Note that meanwhile TLS 1.3 has adopted a similar handshake design.

thus need not be simulated in a security experiment). This $m_4$-message is not required in $\Sigma_0$, since the client Diffie–Hellman share $g^c$ is sent already in message $m_1$ (before $m_2$!), and thus $fin_S$ can be contained in $m_2$.

We stress that one could make TLS-DHE provably secure under DDH by making it more similar to $\Sigma_0$:

1. Include the client Diffie–Hellman share $g^c$ in
   - the first message $m_1$ of TLS-DHE and
   - in the signature sent from client to server.
2. Include all data in $m_4$ of TLS-DHE into message $m_2$ of TLS-DHE, and omit $m_4$.

This modification would allow to carry the security analysis of $\Sigma_0$ from [36] over to TLS-DHE, and thus allow a security proof under DDH instead of PRF-ODH (and the additional standard assumptions on the security of the PRF, the signature scheme, and (considering full TLS-DHE) the stateful encryption scheme).

This would of course require changes to the TLS-DHE protocol, and may therefore be unrealistic. This section should therefore merely be seen as an additional discussion of the issue analysed in this section.

INCLUDING IDENTITIES IN THE CLIENT NONCE Another way (the 'engineering-approach') to circumvent the problem, and thus allow a proof under the DDH assumption instead of PRF-ODH, is to modify TLS such that the client $C$ is able to verify that the server has indeed intended partner $C$, and not some third party $C'$. Note that the client nonce $r_C$ is included in both signatures, in particular in the signature $\sigma_S$ sent from the server to the client. According to [38,41,42], the nonce $r_C$ consists of 28 random bytes (=224 bits). The only requirement on the nonces in the proof is that a collision occurs with

sufficiently small probability, for which 160 bits should be sufficient in practice. One could use the remaining 64 bits to encode an 'identity' that refers uniquely to the client certificate. The server would have to check whether this identity matches the received client certificate. This would, again, require changes to the TLS-DHE protocol and may therefore be unrealistic, even though these changes are minimal.

## 9. Conclusions

In this paper, we have shown that the core cryptographic protocol underlying TLS with ephemeral Diffie–Hellman (DHE) provides a secure establishment of confidential and authenticated channels. 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, if we make the (non-standard) PRF-ODH assumption.

PRF-ODH vs. the Random Oracle model Let us explain what we consider as the main advantage of proofs in the 'standard model with a non-standard assumption' like PRF-ODH over proofs in the random oracle model. A typical proof in the random oracle abstracts a hash function in a *very* strong way, by guaranteeing unconditionally that the hash function has 'essentially all properties required from a good cryptographic hash function' (the formulation is intentionally vague here, because the random oracle model does not clarify precisely which concrete properties are guaranteed: think of programmability, true randomness of outputs, independence of function values from the function input, etc.).

In contrast, a proof in the standard model with a non-standard assumption forces the prover to specify the non-standard properties required from the considered building blocks concretely and in a mathematically precise way. A major advantage of this concreteness is, for example, that it enables researchers to falsify (or verify) whether the concrete instantiation of a cryptographic primitive (like the concrete PRF used in TLS) meets the required specific hardness assumption.

Subsequent and future work on TLS The whole TLS protocol suite is much more complex than the cryptographic protocol underlying TLS-DHE. It is very flexible, as it allows to negotiate cipher suites at the beginning of the TLS Handshake, or to resume sessions using an abbreviated TLS 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 analyse TLS-DHE as a cryptographic protocol. 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 modelling such side channels in complex scenarios like AKE involving many parties and parallel, sequential, and concurrent executions. Another important open problem is to consider cross-protocol attacks that exploit for instance possible subtle connections between different cipher suites. While the example of [89] is impractical, there may be other sophisticated attacks, see [78] for example.

So clearly the security analysis of TLS is not finished yet, there are still many open questions. Subsequent to the conference publication of this paper, major steps towards a better understanding of the security of TLS were made in [12,13,49,61,72], for example. We consider all these results as strong indicators for the soundness of the TLS protocol. We believe that future revisions of the TLS standard should be guided by provable security—ideally in the standard model.

**Acknowledgements**

We would like to thank Dennis Hofheinz, Håkon Jacobsen, Yong Li, Kenny Paterson, Zheng Yang, Hoeteck Wee, and the anonymous reviewers of Crypto 2012 and the Journal of Cryptology for helpful comments and discussions.

## Appendix A: On Choosing the Right Model

Authenticated key exchange (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 want to use a model where *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 [63]. However, explicit authentication is an important security goal for TLS; 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 the authenticity of the channel through which this token was received is crucial.

Second, there is no way to modularize the security proof of TLS in the sense of [35], 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) [31] approach. We think that a formalization in the UC model first requires a thorough analysis in the standard model. 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 [20], it seems in principle possible to model TLS within the UC framework. We refer to [57] for a recent analysis of TLS 1.2 and the current draft of TLS 1.3 in the *constructive cryptography* framework.

On the other hand, we have to make a choice about the enhanced adversarial capabilities newer models offer. We allow for RevealKey queries, but do not take into account RevealState queries. The reason for this omission 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 are erased. So to be precise, we would have to specify several different RevealState queries, which would have added tremendous complexity to both the model and the proof and rendered the paper unreadable.

Thus we have chosen in essence the first model of Bellare and Rogaway [26], adopted to the public-key setting, and enhanced with adaptive corruptions and perfect forward secrecy. Similar variants of this model have been used in [30,35,75], for example.

## References


[1] M. Abdalla, M. Bellare, P. Rogaway, The oracle Diffie–Hellman assumptions and an analysis of DHIES, in *Topics in Cryptology—CT-RSA 2001*, volume 2020 of Lecture Notes in Computer Science, San Francisco, CA, USA, ed. by D. Naccache (Springer, Berlin, Germany, April 8–12, 2001), pp. 143–158

[2] M.R. Albrecht, K.G. Paterson, Lucky microseconds: a timing attack on Amazon's s2n implementation of TLS, in *EUROCRYPT (1) (2016)*, pp. 622–643

[3] N.J. AlFardan, K.G. Paterson, Lucky thirteen: Breaking the TLS and DTLS record protocols, in *2013 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, May 19–22, 2013 (IEEE Computer Society Press, 2013), pp. 526–540

[4] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. Alex Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, Y. Shavitt, DROWN: breaking TLS using sslv2, in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10–12, 2016* (2016), pp. 689–706

[5] G. V. Bard, The vulnerability of SSL to chosen plaintext attack, in *Cryptology ePrint Archive, Report 2004/111* (2004), http://eprint.iacr.org/

[6] G.V. Bard, A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL, in *SECRYPT*, ed. by M. Malek, E. Fernández-Medina, J. Hernando (INSTICC Press, 2006), pp. 99–109

[7] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, J.K. Zinzindohoue, A messy state of the union: taming the composite state machines of TLS, in *2015 IEEE Symposium on Security and Privacy* (IEEE Computer Society Press, 2015), pp. 535–552

[8] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, P.-Y. Strub, Triple handshakes and cookie cutters: breaking and fixing authentication over TLS, in *2014 IEEE Symposium on Security and Privacy* (IEEE Computer Society Press, 2014), pp. 98–113

[9] F. Bergsma, B. Dowling, F. Kohlar, J. Schwenk, D. Stebila, Multi-ciphersuite security of the secure shell (SSH) protocol, in *ACM CCS 14: 21st Conference on Computer and Communications Security* (ACM Press, 2014), pp. 369–381

[10] M. Bellare, New proofs for NMAC and HMAC: security without collision-resistance, in *Advances in Cryptology—CRYPTO 2006*, volume 4117 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by C. Dwork (Springer, Berlin, Germany, August 20–24, 2006), pp. 602–619

[11] K. Bhargavan, C. Fournet, R. Corin, E. Zalinescu, Cryptographically verified implementations for TLS, in *ACM CCS 08: 15th Conference on Computer and Communications Security*, Alexandria, Virginia, USA, ed. by P. Ning, P.F. Syverson, S. Jha (ACM Press, October 27–31, 2008), pp. 459–468

[12] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, Implementing TLS with verified cryptographic security, in *IEEE S&P* (2013), pp. 445–459

[13] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, S.Z. Béguelin, Proving the TLS handshake secure (as it is), in *Advances in Cryptology—CRYPTO 2014, Part II*, volume 8617 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by J.A. Garay, R. Gennaro (Springer, Berlin, Germany, August 17–21, 2014), pp. 235–255

[14] C. Brzuska, M. Fischlin, N.P. Smart, B. Warinschi, S.C. Williams, Less is more: relaxed yet composable security notions for key exchange, *Int. J. Inf. Sec.*, 12(4):267–297, 2013

[15] G. Barthe, B. Grégoire, S. Heraud, S.Z. Béguelin, Computer-aided security proofs for the working cryptographer, in *Advances in Cryptology—CRYPTO 2011*, volume 6841 of Lecture Notes in Computer

Science, Santa Barbara, CA, USA, ed. by P. Rogaway (Springer, Berlin, Germany, August 14–18, 2011), pp. 71–90

[16] C. Brzuska, H. Jacobsen, D. Stebila, Safely exporting keys from secure channels: on the security of EAP-TLS and TLS key exporters, in *EUROCRYPT (1) 2016*, pp. 670–698

[17] M. Bellare, T. Kohno, C. Namprempre, Authenticated encryption in SSH: provably fixing the SSH binary packet protocol, in *ACM CCS 02: 9th Conference on Computer and Communications Security*, Washington D.C., USA, ed. by V. Atluri (ACM Press, November 18–22, 2002), pp. 1–11

[18] M. Bellare, T. Kohno, C. 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

[19] D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1, in *Advances in Cryptology—CRYPTO'98*, volume 1462 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by H. Krawczyk (Springer, Berlin, Germany, August 23–27, 1998), pp. 1–12

[20] B. Barak, Y. Lindell, T. Rabin, *Protocol Initialization for the Framework of Universal Composability*, Cryptology ePrint Archive, Report 2004/006 (2004). http://eprint.iacr.org/

[21] C. Boyd, A. Mathuria, *Protocols for Authentication and Key Establishment*. Information Security and Cryptography (Springer, Berlin, 2003)

[22] C. Badertscher, C. Matt, U. Maurer, P. Rogaway, B. Tackmann, Augmented secure channels and the goal of the TLS 1.3 record layer, in *ProvSec 2015: 9th International Conference on Provable Security*, Lecture Notes in Computer Science (Springer, Berlin, 2015), pp. 85–104

[23] M. Bellare, C. Namprempre, Authenticated encryption: relations among notions and analysis of the generic composition paradigm, in *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of Lecture Notes in Computer Science, Kyoto, Japan, ed. by T. Okamoto (Springer, Berlin, Germany, December 3–7, 2000), pp. 531–545

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

[25] M. Bellare, D. Pointcheval, P. Rogaway, in *Authenticated Key Exchange Secure Against Dictionary Attacks*, in *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of Lecture Notes in Computer Science, Bruges, Belgium, ed. by B. Preneel (Springer, Berlin, Germany, May 14–18, 2000), pp. 139–155

[26] M. Bellare, P. Rogaway, Entity authentication and key distribution, in *Advances in Cryptology—CRYPTO'93*, volume 773 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by D.R. Stinson (Springer, Berlin, Germany, August 22–26, 1994), pp. 232–249

[27] M. Bellare, P. Rogaway, The security of triple encryption and a framework for code-based game-playing proofs, in *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of Lecture Notes in Computer Science, St. Petersburg, Russia, ed. by S. Vaudenay (Springer, Berlin, Germany, May 28–June 1, 2006), pp. 409–426

[28] C. Brzuska, N.P. Smart, B. Warinschi, G.J. Watson, An Analysis of the EMV Channel Establishment Protocol, in *ACM CCS 13: 20th Conference on Computer and Communications Security*, ed. by A.-R. Sadeghi, V. D. Gligor, M. Yung (ACM Press, Berlin, Germany, November 4–8, 2013), pp. 373–386

[29] M. Bellare, B. Tackmann, The multi-user security of authenticated encryption: AES-GCM in TLS 1.3, in *Advances in Cryptology—CRYPTO 2016, Part I*, Lecture Notes in Computer Science, Santa Barbara, CA, USA (Springer, Berlin, Germany, August 2016), pp. 247–276

[30] S. Blake-Wilson, D. Johnson, A. Menezes, Key agreement protocols and their security analysis, in *6th IMA International Conference on Cryptography and Coding*, volume 1355 of Lecture Notes in Computer Science, Cirencester, UK, ed. by M. Darnell (Springer, Berlin, Germany, December 17–19, 1997), pp. 30–45

[31] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in *42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, USA (IEEE Computer Society Press, October 14–17, 2001), pp. 136–145

[32] K.K.R. Choo, C. Boyd, Y. Hitchcock, Examining indistinguishability-based proof models for key establishment protocols, in *Advances in Cryptology—ASIACRYPT 2005*, volume 3788 of Lecture Notes in Computer Science, Chennai, India, ed. by B.K. Roy (Springer, Berlin, Germany, December 4–8, 2005), pp. 585–604

[33] S. Chaki, A. Datta, Aspier: an automated framework for verifying security protocol implementations, in *Computer Security Foundations Symposium, 2009. CSF '09. 22nd IEEE*, (July 2009), pp. 172 –185
[34] J.-S. Coron, M. Joye, D. Naccache, P. Paillier, in *New attacks on PKCS#1 v1.5 encryption* (In Preneel [84]), pp. 369–381
[35] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of Lecture Notes in Computer Science, Innsbruck, Austria, ed. by B. Pfitzmann (Springer, Berlin, Germany, May 6–10, 2001), pp. 453–474
[36] R. Canetti, H. Krawczyk, Security analysis of IKE's signature-based key-exchange protocol, in *Advances in Cryptology—CRYPTO 2002*, volume 2442 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by M. Yung (Springer, Berlin, Germany, August 18–22, 2002), pp. 143–161. http://eprint.iacr.org/2002/120/
[37] C.J.F. Cremers, Session-state reveal is stronger than ephemeral key reveal: attacking the NAXOS authenticated key exchange protocol, in *ACNS 09: 7th International Conference on Applied Cryptography and Network Security*, volume 5536 of Lecture Notes in Computer Science, Paris-Rocquencourt, France, ed. by M. Abdalla, D. Pointcheval, P.-A. Fouque, D. Vergnaud (Springer, Berlin, Germany, June 2–5, 2009), pp. 20–33
[38] T. Dierks, C. Allen, *The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard)*, Obsoleted by RFC 4346, updated by RFCs 3546, 5746 (January 1999)
[39] B. Dowling, M. Fischlin, F. Günther, D. Stebila, A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates, in *ACM CCS 15: 22nd Conference on Computer and Communications Security* (ACM Press, New York, 2015)
[40] B. Dowling, M. Fischlin, F. Günther, D. Stebila, in *A Cryptographic Analysis of the TLS 1.3 Draft-10 Full and Pre-shared Key Handshake Protocol*. Cryptology ePrint Archive, Report 2016/081 (2016). http://eprint.iacr.org/2016/081
[41] T. Dierks, E. Rescorla, in *The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard)*. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746 (April 2006)
[42] T. Dierks, E. Rescorla, in *The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard)* (August 2008), Updated by RFCs 5746, 5878
[43] T. Duong, J. Rizzo, in *The Crime Attack*. https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/ (2012)
[44] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
[45] D. Eastlake III, T. Hansen, in *US Secure Hash Algorithms (SHA and HMAC-SHA)*, RFC 4634 (Informational) (July 2006)
[46] D. Eastlake III, P. Jones, in *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174 (Informational), Updated by RFC 4634 (September 2001)
[47] M. Fischlin, A. Lehmann, D. Wagner, Hash function combiners in TLS and SSL, in *Topics in Cryptology—CT-RSA 2010*, volume 5985 of Lecture Notes in Computer Science, San Francisco, CA, USA, ed. by J. Pieprzyk (Springer, Berlin, Germany, March 1–5, 2010), pp. 268–283
[48] P.-A. Fouque, D. Pointcheval, S. Zimmer, HMAC is a randomness extractor and applications to TLS, in *ASIACCS 08: 3rd Conference on Computer and Communications Security*, Tokyo, Japan, ed. by M. Abe, V. Gligor (ACM Press, March 18–20, 2008), pp. 21–32
[49] F. Giesen, F. Kohlar, D. Stebila, On the security of TLS renegotiation, in *ACM Conference on Computer and Communications Security 2013*, pp. 387–398
[50] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, J. Schwenk, in *Universally composable security analysis of TLS* ProvSec, volume 5324 of LNCS, ed. by J. Baek, F. Bao, K. Chen, X. Lai (Springer, 2008), pp. 313–327
[51] J. Jonsson, B.S. Kaliski Jr, On the security of RSA encryption in TLS, in *Advances in Cryptology—CRYPTO 2002*, pp. 127–142
[52] T. Jager, F. Kohlar, S. Schäge, J. Schwenk, On the security of TLS-DHE in the standard model, in *Advances in Cryptology—CRYPTO 2012*, volume 7417 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by R. Safavi-Naini, R. Canetti (Springer, Berlin, Germany, August 19–23, 2012), pp. 273–293
[53] D. Johnson, A. Menezes, S. Vanstone, The Elliptic Curve Digital Signature Algorithm (ECDSA), *Int. J. Inf. Secur.*, 1(1):36–63, August 2001

[54] T. Jager, J. Schwenk, J. Somorovsky, Practical invalid curve attacks on TLSECDH, in *ACM CCS 15: 22nd Conference on Computer and Communications Security* (ACM Press, New York, 2015), pp. 407–425
[55] T. Jager, J. Schwenk, J. Somorovsky, in *On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS #1 v1.5 Encryption* (ACM CCS 2015), pp. 1185–1196
[56] B. Kaliski, *PKCS #1: RSA Encryption Version 1.5. RFC 2313 (Informational)*, Obsoleted by RFC 2437 (March 1998)
[57] M. Kohlweiss, U. Maurer, C. Onete, B. Tackmann, D. Venturi, in *(De-)Constructing TLS. Cryptology ePrint Archive, Report 2014/020* (2014). http://eprint.iacr.org/
[58] M. Kohlweiss, U. Maurer, C. Onete, B. Tackmann, D. Venturi, (De-)constructing TLS 1.3, in *Progress in Cryptology—INDOCRYPT 2015: 16th International Conference in Cryptology in India*, Lecture Notes in Computer Science (Springer, Berlin, Germany, 2015), pp. 85–102
[59] E. Kiltz, A. O'Neill, A. Smith, Instantiability of RSA-OAEP under chosen-plaintext attack, in *Advances in Cryptology—CRYPTO 2010*, volume 6223 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by T. Rabin (Springer, Berlin, Germany, August 15–19, 2010), pp. 295–313
[60] E. Kiltz, K. Pietrzak, On the security of padding-based encryption schemes—or—why we cannot prove OAEP secure in the standard model, in *Advances in Cryptology—EUROCRYPT 2009*, volume 5479 of Lecture Notes in Computer Science, Cologne, Germany, (Springer, Berlin, Germany, April 26–30, 2009), pp. 389–406
[61] H. Krawczyk, K.G. Paterson, H. Wee, On the security of the TLS protocol: a systematic analysis, in *Advances in Cryptology—CRYPTO 2013, Part I*, volume 8042 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by R. Canetti, J.A. Garay, (Springer, Berlin, Germany, August 18–22, 2013), pp. 429–448
[62] H. Krawczyk, The order of encryption and authentication for protecting communications (or: How secure is SSL?), in *Advances in Cryptology—CRYPTO 2001*, volume 2139 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by J. Kilian, (Springer, Berlin, Germany, August 19–23, 2001), pp. 310–331
[63] H. Krawczyk, HMQV: a high-performance secure Diffie-Hellman protocol, in *Advances in Cryptology—CRYPTO 2005*, volume 3621 of Lecture Notes in Computer Science, Santa Barbara, CA, USA, ed. by V. Shoup (Springer, Berlin, Germany, August 14–18, 2005), pp. 546–566
[64] F. Kohlar, S. Schäge, J. Schwenk, in *On the security of TLS-DH and TLS-RSA in the standard model. Cryptology ePrint Archive, Report 2013/367* (2013). http://eprint.iacr.org/
[65] R. Küsters, M. Tuengerthal, Composition theorems without pre-established session identifiers, in *ACM CCS 11: 18th Conference on Computer and Communications Security*, Chicago, Illinois, USA, ed. by Y. Chen, G. Danezis, V. Shmatikov (ACM Press, October 17–21, 2011), pp. 41–50
[66] H. Krawczyk, H. Wee, The OPTLS protocol and TLS 1.3, in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany* (March 21–24, 2016), pp. 81–96
[67] G. Locke, P. Gallagher, in *FIPS PUB 186-3 Federal Information Processing Standards Publication Digital Signature Standard (DSS)* (2009)
[68] Y. Li, *Personal Communication* (2012)
[69] R. Lychev, S. Jero, A. Boldyreva, C. Nita-Rotaru, How secure and quick is QUIC? Provable security and performance analyses, in *IEEE S&P* (2015 [53]), pp. 214–231
[70] R. Lychev, S. Jero, A. Boldyreva, C. Nita-Rotaru, How secure and quick is QUIC? Provable security and performance analyses, *in Cryptology ePrint Archive, Report 2015/582* (2015). http://eprint.iacr.org/
[71] B.A. LaMacchia, K. Lauter, A. Mityagin, Stronger security of authenticated key exchange, in *ProvSec*, volume 4784 of LNCS, ed. by W. Susilo, J.K. Liu, Y. Mu (Springer, 2007), pp. 1–16
[72] Y. Li, S. Schäge, Z. Yang, F. Kohlar, J. Schwenk, On the security of the pre-shared key ciphersuites of TLS, in *PKC 2014: 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of Lecture Notes in Computer Science, Buenos Aires, Argentina, ed. by H. Krawczyk (Springer, Berlin, Germany, March 26–28, 2014), pp. 669–684
[73] B. Möller, T. Duong, K. Kotowicz, *This Poodle Bites: Exploiting the ssl 3.0 fallback*, PDF online (2014)
[74] J.C. Mitchell, Finite-state analysis of security protocols, in *CAV*, volume 1427 of LNCS, ed. by A.J. Hu, M.Y. Vardi (Springer, 1998), pp. 71–76
[75] P. Morrissey, N.P. Smart, B. Warinschi, A modular security analysis of the TLS handshake protocol, in *Advances in Cryptology—ASIACRYPT 2008*, volume 5350 of Lecture Notes in Computer Science, Melbourne, Australia, ed. by J. Pieprzyk (Springer, Berlin, Germany, December 7–11, 2008), pp. 55–73

[76] P. Morrissey, N.P. Smart, B. Warinschi, The TLS handshake protocol: A modular analysis, *J. Cryptol.*, 23(2):187–223, April 2010
[77] U. Maurer, B. Tackmann, On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption, in *ACM CCS 10: 17th Conference on Computer and Communications Security*, Chicago, Illinois, USA, ed. by E. Al-Shaer, A.D. Keromytis, V. Shmatikov (ACM Press, October 4–8, 2010), pp 505–515
[78] N. Mavrogiannopoulos, F. Vercauteren, V. Velichkov, B. Preneel, A cross-protocol attack on the TLS protocol, in *ACM CCS 12: 19th Conference on Computer and Communications Security*, Raleigh, NC, USA, ed. by T. Yu, G. Danezis, V.D. Gligor (ACM Press, October 16–18, 2012), pp. 62–72
[79] K. Ogata, K. Futatsugi, in *Equational Approach to Formal Analysis of TLS, ICDCS* (IEEE Computer Society, 2005), pp. 795–804
[80] Lawrence C. Paulson. Inductive Analysis of the Internet Protocol TLS. *ACM Trans. Inf. Syst. Secur.*, 2(3):332–351, 1999.
[81] K.G. Paterson, T. Ristenpart, T. Shrimpton, Tag size does matter: attacks and proofs for the TLS record protocol, in *Advances in Cryptology—ASIACRYPT 2011*, volume 7073 of Lecture Notes in Computer Science, Seoul, South Korea, ed. by D.H. Lee, X. Wang (Springer, Berlin, Germany, December 4–8, 2011), pp. 372–389
[82] D. Pointcheval, S. Vaudenay, in *On Provable Security for Digital Signature Algorithms*, Technical report, Ecole Normale Superieure (1996)
[83] M. Ray, S. Dispensa, in *Renegotiating TLS* (2009). http://extendedsubset.com/Renegotiating_TLS
[84] R. Rivest, in *The MD5 Message-Digest Algorithm*. RFC 1321 (Informational) (April 1992)
[85] Q. Sun, D.R. Simon, Y.-M. Wang, W. Russell, V.N. Padmanabhan, L. Qiu, Statistical identification of encrypted web browsing traffic, in *IEEE Symposium on Security and Privacy* (2002), pp. 19–30
[86] J.M. Schanck, W. Whyte, Z. Zhang, Circuit-extension handshakes for Tor achieving forward secrecy in a quantum world, *Proc. Priv. Enhancing Technol.*, 4:219–236, 2016
[87] S. 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 LNCS (2003), pp. 309–323
[88] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, G.M. Masson, Spot me if you can: uncovering spoken phrases in encrypted voip conversations, in *IEEE Symposium on Security and Privacy* (IEEE Computer Society, 2008), pp. 35–49
[89] D. Wagner, B. Schneier, Analysis of the SSL 3.0 protocol, in *Proceedings of the Second USENIX Workshop on Electronic Commerce* (USENIX Association, 1996), pp. 29–40
[90] W. Zeller, E.W. Felten, in *Cross-Site Request Forgeries: Exploitation and Prevention*. Technical report (October 2008). Available at http://from.bz/public/documents/publications/csrf