On the Security of TLS Renegotiation[*]

Florian Giesen[1] Florian Kohlar[1] Douglas Stebila[2]

[1] *Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Bochum, Germany*
florian.giesen@rub.de, florian.kohlar@rub.de

[2] *Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia*
stebila@qut.edu.au

November 7, 2012

Abstract

The Transport Layer Security (TLS) protocol is the most widely used security protocol on the Internet. It supports negotiation of a wide variety of cryptographic primitives through different cipher suites, various modes of client authentication, and additional features such as session resumption and renegotiation. Despite its widespread use, only recently has the full TLS protocol been proven secure, and only then a single ciphersuite family (`TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA`) with no additional features. These additional features have been the cause of practical attacks on TLS. In 2009, Ray and Dispensa demonstrated how TLS renegotiation allows an attacker to splice together its own session with that of a victim, resulting in a man-in-the-middle attack on TLS-reliant applications such as HTTP. TLS was subsequently patched with two defence mechanisms for protection against this attack.

We present the first formal treatment of renegotiation in secure channel establishment protocols. We add optional renegotiation to the authenticated and confidential channel establishment model of Jager et al., an adaptation of the Bellare–Rogaway authenticated key exchange model. We describe the attack of Ray and Dispensa on TLS within our model. Although the two proposed fixes for TLS do not achieve our strongest notion of security, they do achieve a weaker but still reasonable security notion, and TLS can be easily adjusted to achieve that stronger level of security.

Keywords: Transport Layer Security (TLS), renegotiation, security models, key exchange

[*]The research leading to these results has received funding from the European Community (FP7/2007-2013) under grant agreement number ICT-2007-216646 - European Network of Excellence in Cryptology II (ECRYPT II) and the Australian Technology Network–German Academic Exchange Service (ATN-DAAD) Joint Research Co-operation Scheme.

Contents

1 Introduction

Authenticated key exchange (AKE) is an essential cryptographic protocol: it allows two parties to establish a secure (confidential, authenticated, integrity-protected) channel over an insecure (public, unauthenticated, adversary-controlled) connection. AKE protocols have been extensively studied in the literature. There are several mathematical definitions of security for AKE, including the Bellare–Rogaway [BR94] and Blake-Wilson–Johnson–Menezes models [BWJM97] and later variants such as the Canetti–Krawczyk [CK01] and eCK models [LLM07].

The Transport Layer Security (TLS) protocol, the successor of the Secure Sockets Layer (SSL) protocol, provides secure channel establishment on the Internet. It is commonly used to protect information sent via the Hypertext Transfer Protocol (HTTP) on the web, and many other application layer protocols such as email and file transfer. TLS consists of a *handshake* protocol, which is used to agree on security parameters, establish a secret key, and authenticate the parties; and a *record layer* protocol, which is used to send encrypted data.

Despite the importance of TLS, progress on formally modelling the security of TLS has been slow. It turns out that a technicality of TLS prevents it from being proven secure in the standard AKE models listed above: in AKE models, the session key must be indistinguishable from a random key of the same length. However, the final handshake message of the TLS protocol is sent encrypted under the session key. As a result, an adversary can distinguish the session key from a random key by checking the encryption of the final handshake message. Some analyses [JKJ02, MSW08] have shown that a truncated form of the TLS handshake protocol establishes secure keys. Others [GMP$^+$08] deal with a substantially weaker security requirement, namely unauthenticated key agreement. Krawczyk [Kra01] analyzed a variant of the TLS record layer protocol.

Only very recently have analyses of unmodified TLS functionality appeared. Paterson *et al.* at ASIACRYPT 2011 [PRS11] showed that TLS's MAC-then-encode-then-encrypt record layer when used with CBC encryption (with certain length restrictions) satisfies a notion called *length-hiding authenticated encryption (LHAE)*. Subsequently, at CRYPTO 2012 Jager *et al.* [JKSS12] gave the first full proof of the security of (one ciphersuite of) unmodified TLS in a strong security model. Jager *et al.* introduced a variant of the Bellare–Rogaway authenticated key exchange model, called *authenticated and confidential channel establishment (ACCE)*. They proved that the TLS 1.2 protocol using the `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` ciphersuite (which we shorten to `TLS_DHE_DSS_`) is a secure ACCE protocol, under standard assumptions on the cryptographic components. (We note an alternative modular approach to proving the full security of TLS was recently given by Brzuska *et al.* [BFS$^+$12].)

But TLS is not just a basic secure channel: it consists of hundreds of variants with many optional complex functionalities. Alert messages report various error conditions. Previous sessions can be resumed with a shortened handshake. As of May 2012, approximately 300 ciphersuites — combinations of cryptographic primitives — have been standardized. Client authentication is optional, and can be certificate-based or password-based. Various additional options can be specified via extensions and optional fields.

And most importantly for this paper, after a TLS handshake has been completed and transmission on the record layer has started, parties can renegotiate the handshake, to (a) obtain a fresh session key, (b) change cryptographic parameters, or (c) change authentication credentials. For example, if a client needs to authenticate using a client certificate but wishes to not reveal her identity over a public channel, she could first authenticate anonymously (or with pseudonymous credentials), then renegotiate using her real certificate; since the renegotiation messages are transmitted within the existing record layer, the transmission of her certificate is encrypted, and thus she obtains privacy for her identity. We will examine TLS renegotiation in detail, especially in light of previously identified practical attacks related to TLS renegotiation.

Despite the utility of renegotiation in real-world protocols — beyond TLS, renegotiation, rekeying, or reauthentication is also used in the Secure Shell (SSH) protocol, Internet Key Exchange version 2, the Tor anonymity protocol, and others — there has been almost no research in the literature on the security of protocols involving renegotiation, with the exception of a brief note on the TLS renegotiation attack by Farrell [Far10] and the recent thesis of Gelashvili [Gel12], which uses the Scyther tool to automatically identify the TLS renegotiation attack. Rekeying has been studied in the context of group key agreement for applications such as mobile ad hoc networks, but without reference to AKE security models. Renegotiation has been studied in the context of game theoretic protocols, but this does not apply to the provable security

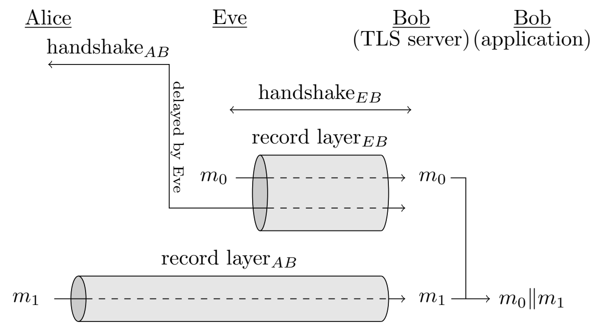


Figure 1: Ray and Dispensa's man-in-the-middle renegotiation attack on TLS-reliant applications

of cryptographic protocols.

1.1 The TLS Renegotiation Issue

All versions of TLS [DA99, DR06, DR08], and SSL v3 [FKK11] before it, support renegotiation. After the initial handshake is completed and secure communication begins in the record layer, either party can request renegotiation. The client can request renegotiation by sending a new `ClientHello` message in the current record layer (i.e., encrypted under the current session key); the server can request renegotiation by sending a `HelloRequest` message in the record layer, which triggers the client to send a new `ClientHello` message.

In November 2009, Ray and Dispensa [RD09] described a man-in-the-middle attack that exploits how certain TLS-reliant applications — such as HTTP over TLS [Res00] — process data across renegotiations. The attack is shown in Figure 1. The attacker Eve observes Alice attempting to establish a TLS session with Bob. Eve delays Alice's initial `ClientHello` and instead establishes her own TLS session with Bob and transmits a message m_0 over that record layer. Then Eve passes Alice's initial `ClientHello` to Bob over the Eve–Bob record layer. Bob views this as a valid renegotiation and responds accordingly; Eve relays the handshake messages between Alice and Bob, who will eventually establish a new record layer to which Eve has no access. Alice then transmits a message m_1 over the Alice–Bob record layer.

This is not strictly speaking an attack on TLS but on how some applications process TLS-protected data. It results from some applications, including HTTPS [RD09] and SMTPS [Zol09], concatenating m_0 and m_1 and treating them as coming from the same party in the same context. For example, if Eve send the HTTP request m_0 and Alice sends the HTTP request m_1, where

$$m_0 = \text{“}\texttt{GET /orderPizza?deliverTo=123-Fake-St} \hookleftarrow \texttt{X-Ignore-This: }\text{”}$$
$$m_1 = \text{“}\texttt{GET /orderPizza?deliverTo=456-Real-St} \hookleftarrow \texttt{Cookie: Account=111A2B}\text{”}$$

(where $\hookleftarrow$ denotes new-line character), then the concatenated request (across multiple lines for readability) is

$$\begin{aligned} m_0\|m_1 = \text{“}&\texttt{GET /orderPizza?deliverTo=123-Fake-St} \hookleftarrow \\ &\texttt{X-Ignore-This: GET /orderPizza?deliverTo=456-Real-St} \hookleftarrow \\ &\texttt{Cookie: Account=111A2B}\text{”} \end{aligned}$$

The "`X-Ignore-This: `" prefix is an invalid HTTP header, and since this header, without a new line character, is concatenated with the first line of Alice's request, the HTTP server ignores the URL that Alice requested and uses the URL that Eve requested. However, the following line, Alice's account cookie, is still processed. Hence Eve has been able to have the pizza delivered to herself but paid for by Alice.

1.2 Countermeasures Added to TLS

The immediate recommendation due to this attack was to disable renegotiation except in cases where it was essential. Subsequently, the Internet Engineering Task Force (IETF) TLS working group developed RFC 5746 [RRDO10] to provide countermeasures to this attack, with the goal of applicability to all versions of TLS and

SSL 3.0. Two countermeasures were standardized: the *Signalling Ciphersuite Value (SCSV)*, designed for older implementations that incorrectly abort when seeing `ClientHello` and `ServerHello` extensions; and the *Renegotiation Information Extension (RIE)*, designed for that support `ClientHello` and `ServerHello` extensions [BWNH+03]. These were adopted by major TLS implementation providers and web browsers and servers, including Apache, Apple, Google, Microsoft, Mozilla, and OpenSSL. A diagram showing the message flow for a generic TLS ciphersuite with SCSV/RIE countermeasures appears in Figure 5 in Appendix E.

Renegotiation Information Extension (RIE). With this countermeasure, each client or server always includes a renegotiation information extension in its respective `ClientHello` or `ServerHello` message. This extension contains one of three values. If the party is not renegotiating, then it includes a fixed "empty" string which denotes that the party supports and understands the renegotiation extension, and the party is in fact not renegotiating. If the party is renegotiating, then it includes the handshake/key confirmation value from the previous handshake: the client sends the previous `client_verify_data` value while the server sends the concatenation of the previous `client_verify_data` and `server_verify_data` values. Intuitively, by including the verify data from the previous handshake, the parties can be assured that they have the same view of the previous handshake, and thus the attack in Figure 1 is avoided.

Signalling Ciphersuite Value (SCSV). SCSV was designed to avoid interoperability problems with TLS 1.0 and SSL 3.0 implementations that did not gracefully ignore extension data at the end of `ClientHello` and `ServerHello` messages. With SCSV, the client uses an alternative method in its initial handshare — an extra, fixed, distinguished ciphersuite value (byte codes `0x00,0xFF`) including in its ciphersuite list — to indicate that it knows how to securely renegotiate. Old servers will ignore this extra value; new servers will recognize that the client supports secure renegotiation, and the server will use the RIE in the remainder of the session. In other words, the only difference between SCSV and RIE is in the `ClientHello` message of the initial handshake: with RIE, the client sends an empty extension, whereas with SCSV the client sends a distinguished value in the list of supported ciphersuites.

1.3 Contributions and Outline

Security model for renegotiable channel establishment protocols. In Section 2, we present a new security model for renegotiable protocols. Since our goal is to analyze the security of TLS, we base our model on that of Jager *et al.* [JKSS12] for *authenticated and confidential channel establishment (ACCE) protocols*, rather than AKE security models, since TLS cannot be proven to be a secure AKE protocol. The primary difference in our model for renegotiable protocols is that each party's oracle (session) can have multiple *phases*; each new phase corresponds to a renegotiation in that session, and can involve the same or different long-term keys. Note that this is qualitatively different than simply having multiple sessions, since short-term values from one phase of a protocol run may be used in the renegotiation for the next phase, where as multiple sessions only reuse long-term values. Each oracle maintains a list of accept/reject, state, and encryption/MAC keys for each phase. Like in TLS, our formalism allows control messages to be sent on the encrypted channel.

The basic goals of a *secure renegotiable ACCE protocol* are that (a) the adversary should not be able to read or inject messages on the encrypted channel, and (b) whenever parties successfully renegotiate, they should have exactly the same view of all previous negotiations and all encrypted messages sent in all previous phases of that session, even when values from previous phases have been compromised.

Analysis of TLS without and with SCSV/RIE countermeasures. Based on the TLS renegotiation attack of Ray and Dispensa, we see in Section 3 that TLS without countermeasures is not secure in our model for renegotiation. We subsequently show in Section 4 that `TLS_DHE_DSS_` with the SCSV or RIE countermeasures of RFC 5746 [RRDO10] is a *weakly secure renegotiable ACCE protocol*, where the adversary is somewhat restricted in the previous secrets she is allowed to reveal; we note that this slightly weaker model is still quite reasonable.

Our approach for proving the renegotiable security of TLS with SCSV/RIE countermeasures is modular. We introduce an intermediate notion, a *secure multi-phase ACCE* (Defn. 4), where there is no link between subsequent phases in a session, and show in Theorem 1 that `TLS_DHE_DSS_` is a secure multi-phase ACCE, meaning that the `client_verify_data` and `server_verify_data` values revealed in the RIE do not weaken the ACCE security of the protocol. We then show in Theorem 2 that any TLS ciphersuite with SCSV/RIE that is a secure multi-phase ACCE is *also* a weakly secure renegotiable ACCE. Although to date only `TLS_DHE_DSS_` has been proven ACCE-secure, it is plausible that many other TLS ciphersuites will be: our generic approach will more easily allow proving the renegotiable ACCE security of TLS ciphersuites. Although ephemeral Diffie–Hellman TLS ciphersuites are not currently as widely used as RSA key transport-based ciphersuites, they are growing in use, for example with Google's 2011 announcement that their default ciphersuite is ephemeral elliptic curve Diffie–Hellman [Lan11]. We also describe how our approach to renegotiation could be extended if and when ciphersuites without forward secrecy, such as RSA key transport, are shown ACCE-like-secure.

New countermeasure for TLS. TLS with SCSV/RIE cannot meet our strongest notion of renegotiable security, only the weaker notion described above. In the strong definition, even if the adversary learns the session key of one phase, parties who later renegotiate still should detect any earlier message injections by the adversary. Though the ability to learn sessions keys of phases while the protocol is still running makes the adversary quite powerful, this may be realistic in scenarios with long-lived session keys, for example with session resumption. We present in Section 5 a simple adjustment to the renegotiation information extension — adding a fingerprint of the transcript of the previous phase's record layer — so TLS can achieve this stronger security notion.

2 Security Definitions for Multi-Phase and Renegotiable ACCE

In this section we describe what a multi-phase authenticated and confidential channel establishment (ACCE) protocol is and our various renegotiation security notions. Our definition builds on the ACCE definition of Jager *et al.* [JKSS12], which combined the Bellare–Rogaway model for authenticated key exchange [BR94] with a stateful variant of Paterson *et al.*'s length-hiding authenticated encryption [PRS11], described in detail in Appendix A.1.

Notation. If S is a set, then $x \xleftarrow{\$} S$ denotes sampling a value x uniformly at random from S. $x \xleftarrow{\$} \mathcal{A}(y)$ denotes the output x of the probabilistic algorithm $\mathcal{A}$ when run on input y and randomly chosen coins. $\mathcal{A}^{\mathcal{O}(\cdot)}$ means $\mathcal{A}$ is run with access to oracle $\mathcal{O}(\cdot)$. The notation $[1, n]$ denotes the set $\{1, 2, \ldots, n\}$; $\texttt{phases}[\ell]$ denotes the ℓth entry in the array `phases` and $|\texttt{phases}|$ denotes the number of entries in the array.

2.1 Overview

The first security notion, a *secure multi-phase ACCE protocol*, is a straightforward extension of the ACCE model to allow multiple, independent phases per session; notably, we require essentially no link between phases:

- An adversary breaks *authentication* if a party accepts in a phase where long-term keys have not been corrupted, but no matching phase exists at the peer.
- An adversary breaks *confidentiality/integrity* if it can guess the bit b involved in a stateful length-hiding authenticated encryption-type confidentiality/integrity experiment.

Our central security definition is that of a *secure renegotiable ACCE protocol*, which strengthens the authentication notion to include renegotiation:

- An adversary breaks *renegotiation authentication* if a party accepts in a phase where long-term keys have not been corrupted, but either no matching phase exists at the peer *or some previous handshake or record layer transcript does not match.*

This captures the idea that parties should successfully renegotiate only when they have exact same view of everything that happened before. We will see in Section 5 that, if TLS when combined with a countermeasure we propose is a secure multi-phase ACCE protocol, then it is also a secure renegotiable ACCE protocol.

However, it is not possible to prove that TLS with the SCSV/RIE countermeasures is a secure renegotiable ACCE protocol: as we will see in Section 3, the strong definition requires that the views of parties match when successfully renegotiating, even when previous sessions' long-term secret keys or session keys were revealed. TLS's SCSV/RIE countermeasures do not fully protect against the case when these secret values are revealed.

As a result, we introduce the weaker, though still quite reasonable, notion of a *weakly secure renegotiable ACCE protocol*, and prove in Section 3 that `TLS_DHE_DSS_` with SCSV/RIE satisfies it:

- An adversary breaks *weak renegotiation authentication* if a party accepts in a phase with uncorrupted long-term keys *and session keys for each earlier phase were not revealed while that phase was active*, but either no matching phase exists at the peer or some previous handshake or record layer transcript does not match.

However, we will also observe that RSA key transport-based ciphersuites do not satisfy this notion: RSA key transport does not offer forward security, so corrupting long-term secret keys is effectively the same as revealing session keys. Note that the ACCE definition of Jager *et al.* [JKSS12] requires forward security, so RSA key transport-based ciphersuites could not be proven ACCE secure. It seems plausible that a non-forward-secure ACCE-like notion could be defined in which RSA key transport-based ciphersuites are secure, and we comment at the end of this section on corresponding renegotiable notions.

We proceed by describing the execution environment for adversaries interacting with multi-phase ACCE protocols, then formally define the various security notions described above.

2.2 Execution Environment

Parties. The environment consists of n_{pa} parties, $\{P_1, \ldots, P_{n_{\mathrm{pa}}}\}$. Each party P_A is a potential protocol participant, and has a list of n_{ke} long-term key pairs $(pk_{A,1}, sk_{A,1}), \ldots, (pk_{A,n_{\mathrm{ke}}}, sk_{A,n_{\mathrm{ke}}})$. We assume that each party P_A is uniquely identified by any one of its public keys $pk_{A,*}$. In practice, there may be other identities that are bound to these public keys, e.g. by using certificates, but this is out of scope of this paper.

Sessions. Each party P_A can participate in up to n_{se} sessions, which are independent executions of the protocol and can be concurrent or subsequent; all of a party's sessions have access to the same list of its long-term key pairs, as well as a trusted list of all parties' public keys. Each session $s \in [1, n_{\mathrm{se}}]$ is presented to the environment as an oracle π_A^s. Each oracle π_A^s records in a variable $\pi_A^s.d$ the oracle corresponding to the intended communication partner, e.g. $\pi_A^s.d = \pi_B^t$.

Phases. Each session can consist of up to n_{ph} phases. Each phase consists of two stages: a *pre-accept*, or "handshake", stage, which is effectively an AKE protocol that establishes a session key and performs mutual authentication; and a *post-accept*, or "record layer", stage, which provides a stateful communication channel with confidentiality and integrity. A list $\pi_A^s.\mathtt{phases}$ of different phase states is maintained; we sometimes use the notation $\pi_A^{s,\ell}$ for $\pi_A^s.\mathtt{phases}[\ell]$. There can be at most n_{ph} phases per oracle. The last entry of $\pi_A^s.\mathtt{phases}$ contains the state of the current phase, which may still be in progress. Each entry $\pi_A^s.\mathtt{phases}[\ell]$ in the log contains:

- pk, the public key used by π_A^s in that phase,
- pk', the public key that π_A^s observed used for the peer in that phase,
- Δ, a counter used to keep track of the current status of the protocol execution,
- α, either `accept`, `reject`, or $\emptyset$ (for in-progress),
- k, the encryption and/or MAC key(s) established by π_A^s in that phase,
- T, the transcript of all messages sent and received by π_A^s during the pre-accept stage of that phase,
- $\rho \in \{\mathsf{Client}, \mathsf{Server}\}$, indicating the role of the oracle,
- RT, the transcript of all ciphertexts sent and received in the post-accept phase by π_A^s encrypted under the key established in that phase,
- b, a random bit sampled by the oracle at the beginning of the phase, and
- st, some additional temporary state (which may, for instance, be used to store ephemeral Diffie–Hellman exponents for the handshake, or state for the sLHAE scheme for the record layer).

The internal state is initialized to $d \leftarrow \emptyset$, $pk \leftarrow \emptyset$, $pk' \leftarrow \emptyset$, $\Delta \leftarrow 1$, $\alpha \leftarrow \emptyset$, $k \leftarrow \emptyset$, $T \leftarrow \emptyset$, $RT \leftarrow \emptyset$, $b \xleftarrow{\$} \{0,1\}$, and $st \leftarrow \emptyset$. When describing a protocol, we will enumerate the protocol messages. The oracles keep track of the protocol execution by setting the counter state equal to the message number that the oracle expects to receive next, and update the counter on each message sent ($\Delta \leftarrow \Delta + 1$). Once a phase of a protocol accepts (that is, an encryption key has been negotiated and authentication is believed to hold), then α is set to `accept`. If the protocol rejects and the oracle wishes to discontinue operation, the counter Δ can be set to the special symbol `reject`. Whenever a new handshake initialization message is received, the oracle adds a new entry to its `phases` list. Application data messages sent and received encrypted under a newly established encryption key (e.g. messages sent in the TLS record layer) will be appended to variable RT in the latest entry of the log. If handshake messages for the renegotiation of a new phase are encrypted under the previous phase's session key (as they are in TLS), those messages are appended to variable T in the new entry of the phase log, not RT in the previous phase.

Remark 1. The introduction of multiple `phases` is the main difference compared to previous AKE and ACCE models. We need to allow multiple authentications and key exchanges within one oracle to capture the functionality of renegotiation. When limited to a single phase and when each party has only one long-term key pair, our execution environment/security experiment is equivalent to the original ACCE model of Jager *et al.* [JKSS12].

Adversarial interaction. The adversary interacts with oracles by issuing the following queries, which allow her to control (forward/alter/create/drop) all communication on the public channel (Send), learn parties' long-term secret keys ($\mathsf{Corrupt}$), learn session keys (Reveal), and control sending and receiving of arbitrary messages on the encrypted record layer ($\mathsf{Encrypt}/\mathsf{Decrypt}$) using a stateful symmetric encryption scheme StE (Appendix A.1).

- $\mathsf{Send}(\pi_A^s, m)$: The adversary can use this query to send any (plaintext) message m of its choosing to (the current phase of) oracle π_A^s. The oracle will respond according to the protocol specification, depending on its internal state. Some distinguished control messages have special behaviour:
  - $m = (\texttt{newphase}, pk)$ triggers an oracle to initiate renegotiation of a new phase (or new session if first phase);
  - $m = (\texttt{ready}, pk)$ activates an oracle to await a new phase/session;

  where for the above control messages, pk indicates the long-term public key pk the oracle should use in the phase; the oracle returns $\perp$ if it does not hold the secret key for pk. Since the control messages do not specify the identity of the peer, this is instead learned during the run of the protocol: we are using a post-specified peer model [CK02]. Delivery of encrypted messages in the post-accept stage are handled by the $\mathsf{Decrypt}$ query below. For protocols such as TLS that perform renegotiation within the encrypted channel, the oracle may reply with an error symbol $\perp$ if it has at least one entry in `phases` and $m \neq (\texttt{newphase}, \cdot)$ or $(\texttt{ready}, \cdot)$.
- $\mathsf{Corrupt}(P_A, pk)$: Oracle π_A^1 responds with the long-term secret key $sk_{A,i}$ corresponding to public key $pk = pk_{A,i}$ of party P_A, or $\perp$ if there is no i such that $pk = pk_{A,i}$. This is the *weak corruption model*, meaning we do not allow the adversary to register rogue public keys.
- $\mathsf{Reveal}(\pi_A^s, \ell)$: Oracle π_A^s responds with the encryption/MAC key(s) for phase ℓ, namely $\pi_A^s.\texttt{phases}[\ell].k$, or $\emptyset$ if no such value exists.
- $\mathsf{Encrypt}(\pi_A^s, ctype, m_0, m_1, len, head)$: This query depends on the random bit b sampled by π_A^s at the beginning of the current phase. It takes as input a content type $ctype$, messages m_0 and m_1, a length len, and header data $head$. Content type `control` is used for handshake messages; in this case, we require that $m_0 = m_1$ to avoid the adversary trivially guessing b by causing different protocol states to emerge, for example by sending a valid handshake message as m_0 and an invalid message as m_1. Content type `data` is used for record layer messages; in this case, one of the two messages (chosen based on bit b) is encrypted for the adversary to distinguish. $\mathsf{Encrypt}$ maintains a counter u initialized to 0 and an encryption state st_e, and proceeds as depicted in Figure 2.
- $\mathsf{Decrypt}(\pi_A^s, C, H)$: This query takes as input a ciphertext C and header data $head$. If π_A^s has not accepted in the current phase, then it returns $\perp$. $\mathsf{Decrypt}$ maintains a counter v and a switch $\mathsf{diverge}$, both initialized to 0, and a decryption state st_d, and proceeds as depicted in Figure 2. If the decryption of C contains a `control` message, then the oracle processes the message according to the protocol specification, which may include updating the state of the oracle and/or creating a new phase, and returns any protocol response message to the adversary, which may or may not be encrypted according to the protocol specification.

```
$\mathsf{Encrypt}(\pi_A^s, ctype, m_0, m_1, len, head)$:
1. $u \leftarrow u + 1$
2. If $(ctype = \texttt{control})$ AND $(m_0 \neq m_1)$, then return $\perp$
3. $(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, len, head, ctype \| m_0, st_e)$
4. $(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, len, head, ctype \| m_1, st_e)$
5. If $(C^{(0)} = \perp)$ OR $(C^{(1)} = \perp)$, then return $\perp$
6. $(C_u, st_e) \leftarrow (C^{(b)}, st_e^{(b)})$
7. Return $C_u$

$\mathsf{Decrypt}(\pi_A^s, C, head)$:
1. $v \leftarrow v + 1$
2. $(ctype \| m, st_d) = \mathsf{StE.Dec}(k, head, C, st_d)$
3. If $(v > u)$ OR $(C \neq C_v)$, then $\mathsf{diverge} \leftarrow 1$
4. If $(b = 1)$ AND $(\mathsf{diverge} = 1)$, then $m' \leftarrow m$
5. If $ctype = \texttt{control}$, then $r' \leftarrow$ protocol response for $m$
6. Else $r' \leftarrow \perp$
7. Return $(m', r')$

where $(k, b, \mathsf{diverge}) = \pi_A^s.\texttt{phases}[\ell^*].(k, b, \mathsf{diverge})$ and $\ell^* = |\pi_A^s.\texttt{phases}|$.
```

Figure 2: Encrypt and Decrypt oracles for the multi-phase/renegotiable ACCE security experiments.

The behaviour of the Decrypt oracle in this combined definition for confidentiality and integrity can be somewhat difficult to understand. It extends that of stateful length-hiding authenticated encryption, for which we give the definition and an explanation in Appendix A.1.

Matching Conversations. Bellare and Rogaway [BR94] introduced the notion of *matching conversations* to help define correctness and security of an AKE protocol.

Definition 1 (Matching conversations)**.** *Let* transcripts T_A *and* T_B *be two sequences of messages sent and received, in chronological order, by parties* P_A *and* P_B *respectively. We say that* T_A *is a* prefix *of* T_B*, if* T_A *contains at least one message, and the messages in* T_A *are identical to and in the same order as the first* $|T_A|$ *messages of* T_B*. We say that transcripts* T_A *and* T_B *are* matching conversations *if (i)* T_B *is a prefix of* T_A *and party* P_A *has sent the last message(s), or (ii)* T_A *is a prefix of* T_B *and party* P_B *has sent the last message(s).*

Remark 2. Matching conversations can also be seen as *post-specified session identifiers.* The asymmetry of the definition — the fact that we have to distinguish which party has sent the last message — is necessary since protocol messages may be sent sequentially. For instance, in the TLS handshake protocol (see Figure 4) the last message the client sends is the 'client finished' message fin_C, and then it waits for the 'server finished' message fin_S before acceptance. The server, however, 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. Since an active adversary may simply drop the last protocol message, we must accommodate this in the definition of matching conversations.

2.3 Security Definitions

In the original security definition for ACCE protocols, security is defined by requiring that (i) the protocol is a secure authentication protocol, thus any party π_A^s reaches the post-accept state only if there exists another party π_B^t such that π_A^s has a matching conversation (in the sense of Def. 1) to π_B^t, and (ii) data transmitted in the post-accept stage over a secure channel is secure (in a sense similar to sLHAE).

We extend this notion to include security when a session has multiple phases that can be renegotiated. We will give several security definitions with different levels of security against renegotiation attacks, as described in the introduction to Section 2.

Each security notion is formally described as a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, with the same overall setup but different winning conditions. In each game, the challenger implements the collection of oracles $\{\pi_A^s : A \in [1, n_{\text{pa}}], s \in [1, n_{\text{se}}]\}$. At the beginning of the game, the challenger generates n_{ke} long-term key pairs $(pk_{A,1}, sk_{A,1}), \ldots, (pk_{A,n_{\text{ke}}}, sk_{A,n_{\text{ke}}})$ for each party P_A; we assume that, within a party, all public key pairs are distinct. (That distinct parties have distinct key pairs comes as a consequence of the protocol being secure.) The adversary receives all parties' public keys as input. The adversary may issue Send, Corrupt, Reveal, Encrypt, and Decrypt queries to the oracles; recall that Send allows the party to control which public keys are used in which phases and when phases are renegotiated. Finally, the adversary outputs a bit b' and terminates.

Table 1 at the end of the section provides a comparative summary of the various security notions introduced in this section, as well as a summary of the results on TLS that appear in the rest of this paper.

Definition 2 (Correct multi-phase ACCE)**.** *We say* Π *is a* correct multi-phase ACCE protocol *if, for all oracles* π_A^s *with destination address* $\pi_A^s.d = \pi_B^t$*, and for all* $\ell, \ell' \in [1, n_{\mathrm{ph}}]$ *for which* $\pi_A^s.\mathtt{phases}[\ell].T$ *is a matching conversation to* $\pi_B^t.\mathtt{phases}[\ell'].T$*, it holds that* $\pi_A^s.\mathtt{phases}[\ell].\alpha = \mathtt{accept}$*.*

2.3.1 Confidentiality.

All of our notions for secure ACCE protocols will require confidentiality/integrity of the post-accept stage record layer in each uncorrupted phase. Intuitively, an adversary should not be able to guess the bit b used in the Encrypt/Decrypt oracles in a phase where she has not impersonated the parties (i.e., corrupted the long-term secret keys before the phase accepted) or revealed the session key of the party or its peer. As with the ACCE notion of Jager *et al.* [JKSS12], this notion ensures forward security: corrupting long-term secret keys after completion of a session should not impact confidentiality/integrity of messages.

Definition 3 (Confidentiality/integrity)**.** *Suppose an algorithm* $\mathcal{A}$ *with running time* τ *interacts with a multi-phase ACCE protocol* Π *in the above execution environment and returns a tuple* (A, s, ℓ, b')*. If*

C1. $\pi_A^s.\mathtt{phases}[\ell].\alpha = \mathtt{accept}$*; and*
C2. $\mathcal{A}$ *did not query* $\mathsf{Corrupt}(P_A, \pi_A^s.\mathtt{phases}[\ell].pk)$ *before* π_A^s *accepted in phase* ℓ*; and*
C3. $\mathcal{A}$ *did not query* $\mathsf{Corrupt}(P_B, \pi_A^s.\mathtt{phases}[\ell].pk')$ *before* π_A^s *accepted in phase* ℓ*, where* $\pi_A^s.d = \pi_B^t$*; and*
C4. $\mathcal{A}$ *did not query* $\mathsf{Reveal}(\pi_A^s, \ell)$*; and*
C5. $\mathcal{A}$ *did not query* $\mathsf{Reveal}(\pi_B^t, \ell')$*, where* $\pi_B^t = \pi_A^s.d$ *is* π_A^s*'s intended communication partner, and* ℓ' *is any phase for which* $\pi_B^t.\mathtt{phases}[\ell'].T$ *is a matching conversation to* $\pi_A^s.\mathtt{phases}[\ell].T$*; and*
C6. $|\Pr[\pi_A^s.\mathtt{phases}[\ell].b = b'] - 1/2| \geq \epsilon$*,*

then we say that $\mathcal{A}$ (τ, ϵ)-breaks confidentiality/integrity of Π.

2.3.2 Secure multi-phase ACCE.

First we state a straightforward extension of the ACCE model to protocols with multiple phases, but with essentially no security condition relating one phase to another. This definition captures the properties of TLS without any renegotiation countermeasures, and will be used in our generic result in Section 4. Intuitively, for this simplest notion of authentication, an adversary should not be able to cause a phase to accept unless there exists a phase at the peer with a matching pre-accept handshake transcript, provided she has not impersonated the parties (i.e., corrupted the long-term secret keys before the phase accepted).

Definition 4 (Secure multi-phase ACCE)**.** *Suppose an algorithm* $\mathcal{A}$ *with running time* τ *interacts with a multi-phase ACCE protocol* Π *in the above execution environment and terminates. If, with probability at least* ϵ*, there exists an oracle* π_A^s *with* $\pi_A^s.d = \pi_B^t$ *and a phase* ℓ *such that*

A1. $\pi_A^s.\mathtt{phases}[\ell].\alpha = \mathtt{accept}$*; and*
A2. $\mathcal{A}$ *did not query* $\mathsf{Corrupt}(P_A, \pi_A^s.\mathtt{phases}[\ell].pk)$ *before* π_A^s *accepted in phase* ℓ*; and*
A3. $\mathcal{A}$ *did not query* $\mathsf{Corrupt}(P_B, \pi_A^s.\mathtt{phases}[\ell].pk')$ *before* π_A^s *accepted in phase* ℓ*, where* $\pi_A^s.d = \pi_B^t$*; and*
M. *there is no* ℓ' *such that* $\pi_B^t.\mathtt{phases}[\ell'].T$ *is a matching conversation to* $\pi_A^s.\mathtt{phases}[\ell].T$

then we say that $\mathcal{A}$ (τ, ϵ)-breaks authentication of Π.

A protocol Π *is a* (τ, ϵ)-secure multi-phase ACCE protocol *if there exists no algorithm* $\mathcal{A}$ *that* (τ, ϵ)*-breaks confidentiality/integrity (Def. 3) or authentication (as defined above) of* Π*.*

2.3.3 Secure renegotiable ACCE.

We next strengthen the authentication notion to include renegotiation. Intuitively, an adversary should not be able to cause a phase to accept unless there exists a phase at the peer with a matching pre-accept handshake transcript and all previous phases' handshake and record layer transcripts match, provided she has not impersonated the parties in the current phase. We will show in Section 5 that `TLS_DHE_DSS_` with our proposed countermeasure satisfies this definition.

Definition 5 (Secure renegotiable ACCE)**.** *Suppose an algorithm* $\mathcal{A}$ *with running time* τ *interacts with a multi-phase ACCE protocol* Π *in the above execution environment and terminates. If, with probability at least* ϵ*, there exists an oracle* π_A^s *with* $\pi_A^s.d = \pi_B^t$ *and a phase* ℓ^* *such that*

A1–A3 *as in Definition 4 with ℓ^*, and either*
M′(a) $\pi_B^t.\mathtt{phases}[\ell^*].T$ *is not a matching conversation to* $\pi_A^s.\mathtt{phases}[\ell^*].T$ *or*
M′(b) *for some* $\ell < \ell^*$, $\pi_A^s.\mathtt{phases}[\ell].T \| RT \neq \pi_B^t.\mathtt{phases}[\ell].T \| RT$*;*

then we say that $\mathcal{A}$ (τ, ϵ)-breaks renegotiation authentication of Π.

A protocol Π *is a* (τ, ϵ)-secure renegotiable ACCE protocol *if there exists no algorithm* $\mathcal{A}$ *that* (τ, ϵ)*-breaks confidentiality/integrity (Def. 3) or renegotiation authentication (as defined above) of* Π.

2.3.4 Weakly secure renegotiable ACCE.

Unfortunately, the `TLS_DHE_DSS_` ciphersuite, when combined with SCSV/RIE countermeasures, does not satisfy Def. 5 because, as we will see in Section 4.1, revealing session keys in earlier phases allows the adversary to change the messages on the record layer in earlier phases, but the SCSV/RIE countermeasure will not detect this.

Of course, revealing earlier phases' session keys while that phase is active and still expecting detection when renegotiating later is a strong security property, and the lack of this property does not imply an attack in most scenarios. This motivates a slightly weaker renegotiation notion: when previous phases' session keys are not revealed while that phase is active and the current phase's long-term secret keys are not corrupted, the adversary should not be able to cause a phase to accept unless there exists a phase at the peer with a matching pre-accept handshake transcript and all previous phases' handshake and record layer transcripts match.

Definition 6 (Weakly secure renegotiable ACCE)**.** *Suppose an algorithm* $\mathcal{A}$ *with running time* τ *interacts with a multi-phase ACCE protocol* Π *in the above execution environment and terminates. If, with probability at least* ϵ*, there exists an oracle* π_A^s *with* $\pi_A^s.d = \pi_B^t$ *and a phase* ℓ^* *such that all conditions from Def. 5, as well as the following additional conditions are satisfied:*

A4. $\mathcal{A}$ *did not issue a* $\mathsf{Reveal}(\pi_A^s, \ell)$ *query before* π_A^s *accepted in phase* $\ell + 1$*, for every* $\ell < \ell^*$*, and*
A5. $\mathcal{A}$ *did not issue a* $\mathsf{Reveal}(\pi_B^t, \ell)$ *query before* π_A^s *accepted in phase* $\ell + 1$*, for every* $\ell < \ell^*$*;*

then we say that $\mathcal{A}$ (τ, ϵ)-breaks weak renegotiation authentication of Π.

A protocol Π *is a* (τ, ϵ)-weakly secure renegotiable ACCE protocol *if there exists no algorithm* $\mathcal{A}$ *that* (τ, ϵ) *breaks confidentiality/integrity (Def. 3) or weak renegotiation authentication (as defined above) of* Π.

Remark 3. While conditions **A4** and **A5** prohibit the adversary from revealing encryption keys of previous phases while active *for the purposes of breaking authentication*, the confidentiality/integrity aspect of Def. 6 still places no such restriction on previous encryption keys being revealed.

2.3.5 Protocols without forward security.

The ACCE notion of Jager *et al.* [JKSS12] requires confidentiality/integrity of ciphertexts to hold even when the long-term secret keys are corrupted after the handshake completes. As a result, RSA key transport-based ciphersuites cannot be proven secure in this model: they do not have forward security, since corrupting the long-term secret keys allows the adversary to compute the session key.

A non-forward-secure notion of renegotiable ACCE could be defined in which RSA key transport could be proven secure. For renegotiation authentication, public keys used in phase ℓ should not be $\mathsf{Corrupt}$ed while the post-accept stage of phase ℓ is active, as the adversary could compute the session key and inject messages undetectably. For confidentiality/integrity, public keys used in phase ℓ should never be $\mathsf{Corrupt}$ed, as the adversary could compute the session key and distinguish ciphertexts. A more detailed discussion appears in Appendix B.1.

3 Renegotiation (In)security of TLS

In this section we discuss how the original TLS protocol, without SCSV/RIE countermeasures, fits into our model, and show how the attack of Ray and Dispensa is captured in the model. A discussion on TLS session resumption appears in Appendix C.

	Secure multi-phase ACCE (Defn. 4)	**Weakly secure renegotiable ACCE** (Defn. 6)	**Secure renegotiable ACCE** (Defn. 5)
Secure against Ray–Dispensa-type attack	×	✓ with query restrictions **A5**,**A6**	✓
Authentication			
A2. Corrupt *pk* before acceptance	not allowed	not allowed	not allowed
A3. Corrupt peer's *pk* before acceptance	not allowed	not allowed	not allowed
A4. Reveal session keys of previous phases while active	allowed	not allowed	allowed
A5. Reveal session keys of previous phases while active	allowed	not allowed	allowed
M. every phase that accepts has a matching handshake transcript at *some* phase of the peer	implied		
M′(a) every phase that accepts has a matching handshake transcript at *the same* phase of the peer		implied	implied
M′(a) when a phase accepts, handshake and record layer transcripts in *all previous phases* equal those at the peer		implied	implied
Confidentiality/integrity (Defn. 3)	implied	implied	implied
`TLS_*` without countermeasures	—	× (Sect. 3.1)	× (Sect. 3.1)
`TLS_RSA_` with SCSV/RIE countermeasures	?[1]	× (App. B.2) / ?[1]	× (Sect. 4.1)
`TLS_DHE_DSS_` with SCSV/RIE countermeasures	✓ (Thm. 1)	✓ (Cor. 1)	× (Sect. 4.1)
`TLS_RSA_` with new (Sect. 5) countermeasure	?[1]	× (App. B.2) / ?[1]	× (App. B.2) / ?[1]
`TLS_DHE_DSS_` with new (Sect. 5) countermeasure	✓ (Thm. 3)	✓ (Thm. 3)	✓ (Thm. 3)

Table 1: Summary of security notions and results on TLS

[1] `TLS_RSA_` key transport ciphersuites may be able to be shown secure under notions with suitable restrictions on forward security; see discussion at end of Sect. 2.3.5 and Appendix B.

Jager *et al.* [JKSS12] in the full version [JKSS11, Fig. 3] described how to map TLS into the ACCE model. We highlight a few components of that mapping, and the alterations due to our addition of renegotiation.

Oracles generally respond to Send, Encrypt, and Decrypt queries as specified by the TLS handshake and record layer protocols. The Send control message $m = (\texttt{newphase}, pk)$ when sent to a client causes the client to send a new `ClientHello` message, and when sent to a server causes the server to send a new `HelloRequest` message. For the Encrypt and Decrypt queries, we use a content type field *ctype* that corresponds to the `ContentType` field of the `TLSPlaintext` datatype in the TLS record layer specification [DR08, §6.2.1]: packets with `ContentType`=`change_cipher_spec (20)` or `handshake (22)` are considered in our model to have *ctype* = `control` and packets with `ContentType`=`application_data (23)` are considered in our model to have *ctype* = `data`. We do not explicitly handle `ContentType`=`alert (21)` messages. The Reveal query reveals the encryption and MAC keys derived from the master secret key, *not* the master secret key itself.

3.1 TLS without countermeasures is not a (weakly) secure renegotiable ACCE protocol

Recall the TLS renegotiation attack by Ray and Dispensa [RD09], as described previously in Figure 1. The attacker Eve observes Alice attempting to establish a TLS session with Bob. Eve delays Alice's initial `ClientHello` and instead establishes her own TLS session with Bob and transmits a message m_0 over that record layer. Then Eve passes Alice's initial `ClientHello` to Bob over the Eve–Bob record layer. Bob views this as a valid renegotiation and responds accordingly; Eve relays the handshake messages between Alice and Bob, who will eventually establish a new record layer to which Eve has no access. Alice then transmits a message m_1 over the Alice–Bob record layer. Intuitively, this is a valid attack: Alice believes this is the initial handshake, but Bob believes this is a renegotiated handshake.

Formally, this attack is captured in our weakly secure renegotiable ACCE model of Definition 6 as follows. Assume Alice and Bob each have a single oracle instance, and Eve has carried out the above attack. Then for Bob's oracle π^1_{Bob}, the value of ℓ^* is 2: the last entry in `phases` where Bob has a matching handshake transcript to some handshake transcript in Alice's oracle π^1_{Alice} is the second (and last) `phases` entry. The attacker has broken renegotiation authentication at both Alice and Bob's instances. At Alice by satisfying condition $\mathbf{M'(a)}$ (Alice's first handshake transcript does not match Bob's first handshake transcript), and at Bob by satisfying both $\mathbf{M'(a)}$ (Bob's second handshake transcript does not match Alice's second handshake transcript) and $\mathbf{M'(b)}$ (for every $\ell < 2$, Bob's ℓth handshake and record layer transcript does not match Alice's ℓth transcripts). Thus TLS without countermeasures is not a weakly secure or secure renegotiable ACCE.

4 Renegotiation Security of TLS with SCSV/RIE Countermeasures

In this section we analyze the security of TLS with the SCSV/RIE countermeasures proposed in RFC 5746 [RRDO10]. We first see that the SCSV/RIE countermeasures are not enough to prove that TLS satisfies our strongest notion, a secure renegotiable ACCE (Defn. 5), because revealing previous phases' session keys allows an attacker to manipulate the record layer and still have renegotiation succeed.

We will show our central result, that `TLS_DHE_DSS_` is a weakly secure renegotiable ACCE, first by showing that it is a secure multi-phase ACCE (Thm. 1) and then combining with a generic result (Thm. 2) that says that any TLS ciphersuite that is a secure multi-phase ACCE when using SCSV/RIE is also a weakly secure renegotiable ACCE when using SCSV/RIE.[1] Even with SCSV/RIE, RSA key transport-based ciphersuites are not weakly secure renegotiable ACCE protocols as they do not have forward secrecy (see Appendix B.2).

4.1 `TLS_*` with SCSV/RIE is not a secure renegotiable ACCE

Definition 5 requires that, even when the adversary can reveal previous phases' session keys, the parties will not successfully renegotiate if the attacker has manipulated the record layer. The SCSV/RIE countermeasures do not protect against this type of adversary. They only provide assurance that handshake transcripts from previous phases match exactly. TLS itself of course provides integrity protection for record layer transcripts via the message authentication codes, but Definition 5 allows the adversary to reveal the encryption and MAC keys of previous phases. Thus, an adversary who reveals the current encryption and MAC keys can modify record layer messages but Alice and Bob will still successfully renegotiate a new phase (although the adversary must not alter the number of messages sent, as the *number* of record layer messages sent in the previous phase happens to be protected by SCSV and RIE countermeasures).

We emphasize that while this demonstrates a theoretical weakness in TLS renegotiation countermeasures compared to our very strong security model, it does not translate into an attack on TLS renegotiation countermeasures when intermediate phases' encryption and MAC keys are not revealed.

4.2 `TLS_DHE_DSS_` with SCSV/RIE is a secure multi-phase ACCE

We now aim to prove that the SCSV/RIE renegotiation countermeasures introduced in TLS are secure. We will do so via two results. First, we will show that the specific ciphersuite `TLS_DHE_DSS_` with renegotiation information extensions (RIE) (shown in Figure 4) is a secure multi-phase ACCE protocol, meaning that the inclusion of the RIE does not leak any damaging information. Then we will show generically that any TLS protocol with RIE that is a secure multi-phase ACCE is also a weakly secure renegotiable ACCE protocol.

The theorem relies on the *PRF-Oracle-Diffie–Hellman* (PRFODH) *assumption* as used by Jager *et al.* [JKSS12], the definition of which appears in Appendix A.2.

Theorem 1 (`TLS_DHE_DSS_` with SCSV/RIE is a secure multi-phase ACCE)**.** *Let μ be the output length of* PRF *and let λ be the length of the nonces r_C and r_S. Assume that the pseudo-random function* PRF *is $(\tau, \epsilon_{\mathsf{prf}})$-secure, the signature scheme is $(\tau, \epsilon_{\mathsf{sig}})$-secure, the* DDH*-problem is $(\tau, \epsilon_{\mathsf{ddh}})$-hard in the group G used to compute the TLS premaster secret, and the* PRFODH*-problem is $(\tau, \epsilon_{\mathsf{prfodh}})$-hard with respect to G and* PRF*. Suppose that the stateful symmetric encryption scheme is $(\tau, \epsilon_{\mathsf{sLHAE}})$-secure.*

[1]It would be nice to prove that any secure ACCE protocol plus SCSV/RIE becomes a weakly secure renegotiable ACCE, but we need to make the intermediate step through multi-phase ACCE to ensure that SCSV/RIE does not leak anything important in that ciphersuite.

For any adversary that $(\tau', \epsilon_{\mathsf{tls}})$-breaks `TLS_DHE_DSS_` *in the sense of Definition 4 with $\tau \approx \tau'$ it holds that*

$$\epsilon_{\mathsf{tls}} \leq 4n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(n_{\mathrm{pa}}n_{\mathrm{ke}}\epsilon_{\mathsf{sig}} + \frac{5}{4}\epsilon_{\mathsf{ddh}} + \frac{5}{2}\epsilon_{\mathsf{prf}} + \frac{1}{4}\epsilon_{\mathsf{sLHAE}} + n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}}\right)\right).$$

Recall that $n_{\mathrm{pa}}, n_{\mathrm{se}}, n_{\mathrm{ph}}$, and n_{ke} are the maximum number of parties, sessions per party, phases per session, and keypairs per party, respectively.

The proof of Theorem 1 follows closely the proof of Jager *et al.* [JKSS12, Thm. 2] that `TLS_DHE_DSS_` is a secure ACCE protocol. The two key differences are as follows. First, we need to account for the additional key exchanges that occur in multiple phases. Second, we need to ensure that including the RIE, which contains finished values of previous phases, does not affect the security of the current phase. However, Jager *et al.* have proved that a truncated `TLS_DHE_DSS_` protocol, in which the Finished messages are sent in plain, provide indistinguishable session keys. In the proof, we exchange the PRF used to derive the application keys with a truly random function, thus the output does not leak any information about the input.

To prove Theorem 1, we follow the approach of Jager *et al.* [JKSS11] and divide the set of all adversaries into two categories:

1. Adversaries that succeed in making an oracle accept maliciously. We call such an adversary an *authentication-adversary.*
2. Adversaries that do not succeed in making any oracle accept maliciously, but which answer the encryption/integrity challenge. We call such an adversary an *encryption-adversary.*

We prove Theorem 1 by the following two lemmas. Lemma 1 bounds the probability ϵ_{auth} that an authentication-adversary succeeds, Lemma 2 bounds the probability ϵ_{enc} that an encryption-adversary succeeds. Then we have

$$\epsilon_{\mathsf{tls}} \leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{enc}} \ .$$

Lemma 1. *For any adversary running in time $\tau' \approx \tau$, the probability that there exists an oracle π_i^s that accepts maliciously is at most*

$$\epsilon_{\mathsf{auth}} \leq 2n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(n_{\mathrm{pa}}n_{\mathrm{ke}}\epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}}\right)\right)$$

where all quantities are defined as stated in Theorem 1.

Note that $\epsilon_{\mathsf{auth}} \leq \epsilon_{\mathsf{client}} + \epsilon_{\mathsf{server}}$, where $\epsilon_{\mathsf{client}}$ is an upper bound on the probability that there exists an oracle with $\rho = \mathsf{Client}$ that accepts maliciously in the sense of Definition 4, and $\epsilon_{\mathsf{server}}$ is an upper bound on the probability that there exists an oracle with $\rho = \mathsf{Server}$ that accepts maliciously. We claim that

$$\epsilon_{\mathsf{client}} \leq n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(n_{\mathrm{pa}}n_{\mathrm{ke}}\epsilon_{\mathsf{sig}} + n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}}\right)\right)$$
$$\epsilon_{\mathsf{server}} \leq n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(\frac{n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}}{2^{\lambda}} + n_{\mathrm{pa}}n_{\mathrm{ke}}\epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}}\right)$$

and thus

$$\begin{aligned}\epsilon_{\mathsf{auth}} &\leq \epsilon_{\mathsf{client}} + \epsilon_{\mathsf{server}} \\ &\leq 2n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(n_{\mathrm{pa}}n_{\mathrm{ke}}\epsilon_{\mathsf{sig}} + \epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + n_{\mathrm{pa}}n_{\mathrm{se}}n_{\mathrm{ph}}\left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}}\right)\right).\end{aligned}$$

4.2.1 Proof of Lemma 1: $\epsilon_{\mathsf{client}}$

Proof. We first show, that the probability that there exists an oracle with $\rho = \mathsf{Client}$ that accepts maliciously in the sense of Definition 4 is negligible. The proof proceeds in a *sequence of games*, following [BR06, Sho04]. The first game is the real security experiment. We then describe several intermediate games that modify the original game step-by-step, and argue that our complexity assumptions imply that each game is computationally indistinguishable from the previous one. We end up in the final game, where no adversary can break the security of the protocol.

Let $\mathsf{break}_{\delta}^{(1)}$ be the event that occurs when the first oracle that accepts maliciously in the sense of Definition 4 with $\rho = \mathsf{Client}$ in Game δ.

Game 0. This game equals the multi-phase ACCE security experiment used in Section 2.2. Thus, for some $\epsilon_{\mathsf{client}}$ we have

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

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

In total less than $n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}}$ 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 $(n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}})^2 2^{-\lambda}$, which implies

$$\Pr[\mathsf{break}_0^{(2)}] \le \Pr[\mathsf{break}_1^{(2)}] + \frac{(n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}})^2}{2^\lambda} \ .$$

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 and the phase in which this happens. If our guess is wrong, i.e., if there is another (Client or Server) oracle that accepts before or if they accept in a different phase, then we abort the game.

Technically, this game is identical, except for the following. The challenger guesses three random indices $(i^*, s^*, \ell^*) \xleftarrow{\$} [1, n_{\mathrm{pa}}] \times [1, n_{\mathrm{se}}] \times [1, n_{\mathrm{ph}}]$. If there exists an oracle π_i^s that accepts maliciously in phase ℓ, and $(i, j, \ell) \neq (i^*, j^*, \ell^*)$ and π_i^s has $\rho \neq \mathsf{Client}$, then the challenger aborts the game. Note that if the first oracle π_i^s that accepts maliciously has $\rho = \mathsf{Client}$, then with probability $1/(n_{\mathrm{pa}} \cdot n_{\mathrm{se}} \cdot n_{\mathrm{ph}})$ we have $(i, j, \ell) = (i^*, j^*, \ell^*)$, and thus

$$\Pr[\mathsf{break}_1^{(2)}] = n_{\mathrm{ph}} n_{\mathrm{pa}} n_{\mathrm{se}} \Pr[\mathsf{break}_2^{(2)}] \ .$$

Note that in this game the attacker can only break the security of the protocol if oracle $\pi_{i^*}^{s^*}$ is the first oracle that accepts maliciously in phase ℓ and has $\rho = \mathsf{Client}$; 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 $\pi_{i^*}^{s^*,\ell^*}$ 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 $\pi_{i^*}^{s^*,\ell^*}$ as first input (note that there may be several such oracles, since the attacker may send copies of r_C to many oracles).

Technically, we abort and raise event $\mathsf{abort}_{\mathsf{sig}}$, if oracle $\pi_{i^*}^{s^*,\ell^*}$ 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 σ_S is a valid signature over $r_C \| r_S \| p \| g \| T_S$, but there exists no oracle π_j^t which has previously output σ_S. Clearly we have

$$Pr[\mathsf{break}_2^{(1)}] \le \Pr[\mathsf{break}_3^{(1)}] + \Pr[\mathsf{abort}_{\mathsf{sig}}] \ .$$

Note that the experiment is aborted, if $\pi_{i^*}^{s^*,\ell^*}$ does not accept *maliciously*, due to Game 2. This means that party P_j must not be corrupted when $\pi_{i^*}^{s^*,\ell^*}$ accepts (as otherwise $\pi_{i^*}^{s^*,\ell^*}$ does not accept *maliciously*). To show that $\Pr[\mathsf{abort}_{\mathsf{sig}}] \le n_{\mathrm{ke}} n_{\mathrm{pa}} \cdot \epsilon_{\mathsf{sig}}$, we construct a signature forger as follows. The forger receives as input a public key pk^* and simulates the challenger for $\mathcal{A}$. It guesses two indices $\phi_1 \xleftarrow{\$} [1, n_{\mathrm{pa}}]$ and $\phi_2 \xleftarrow{\$} [1, n_{\mathrm{ke}}]$, sets $pk_{\phi_1,\phi_2} = pk^*$, 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_{ϕ_1,ϕ_2} when necessary.

If $\phi_1 = j$ and the corresponding public key is pk_{j,ϕ_2}, which happens with probability $1/(n_{\mathrm{pa}} n_{\mathrm{ke}})$, then the forger can use the signature received by $\pi_{i^*}^{s^*,\ell^*}$ to break the EUF-CMA security of the signature scheme with success probability ϵ_{sig}. Therefore we gain that $\Pr[\mathsf{abort}_{\mathsf{sig}}]/(n_{\mathrm{pa}} n_{\mathrm{ke}}) \le \epsilon_{\mathsf{sig}}$; if $\Pr[\mathsf{abort}_{\mathsf{sig}}]$ is not negligible, then ϵ_{sig} is not negligible as well and we have

$$Pr[\mathsf{break}_2^{(1)}] \le \Pr[\mathsf{break}_3^{(1)}] + n_{\mathrm{pa}} n_{\mathrm{ke}} \epsilon_{\mathsf{sig}} \ .$$

Note that in Game 3 oracle $\pi_{i^*}^{s^*,\ell^*}$ receives as input a Diffie–Hellman value T_S such that T_S was chosen by another oracle, but not by the attacker. Note also that there may be multiple oracles that issued a signature σ_S containing r_C, since the attacker may have sent several copies of r_C to several oracles.

Game 4. In this game we want to make sure that we know the oracle π_j^t (and appropriate phase ℓ_t) which will issue the signature σ_S that $\pi_{i^*}^{s^*}$ receives in phase ℓ^*. Note that this signature includes the random nonce r_S, which is unique due to Game 1. Therefore the challanger in this game proceeds as before, but additionally guesses three indices $(j^*, t^*, \ell_t^*) \xleftarrow{\$} [1, n_{\mathrm{pa}}] \times [1, n_{\mathrm{se}}] \times [1, n_{\mathrm{ph}}]$. It aborts, if the attacker does *not* make a Send-query containing r_C to $\pi_{j^*}^{t^*,\ell_t^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ responds in this phase with messages containing σ_S such that σ_S is forwarded to $\pi_{i^*}^{s^*,\ell^*}$.

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

$$\Pr[\mathsf{break}_3^{(1)}] \leq n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \Pr[\mathsf{break}_4^{(1)}] \ .$$

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

Game 5. Recall that $\pi_{i^*}^{s^*,\ell^*}$ 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 $\pi_{j^*}^{t^*,\ell_t^*}$, and t_c denotes the Diffie–Hellman exponent chosen by $\pi_{i^*,\ell_t^*}^{s^*}$. In this game we replace the master secret ms computed by $\pi_{i^*,\ell_t^*}^{s^*}$ with an independent random value $\widetilde{ms}$. Moreover, if $\pi_{j^*}^{t^*,\ell_t^*}$ receives as input the same Diffie–Hellman share T_C that was sent from $\pi_{i^*}^{s^*,\ell^*}$, then we set the master secret of $\pi_{j^*}^{t^*,\ell_t^*}$ equal to $\widetilde{ms}$. Otherwise we compute the master secret as specified in the protocol. We claim that

$$Pr[\mathsf{break}_4^{(1)}] \leq \Pr[\mathsf{break}_5^{(1)}] + \epsilon_{\mathsf{PRFODH}} \ .$$

Suppose there exists an adversary $\mathcal{A}$ that distinguishes Game 5 from Game 4. We show that this implies an adversary $\mathcal{B}$ that solves the PRFODH problem.

Adversary $\mathcal{B}$ outputs $(label_1 \| r_C \| r_S)$ to its oracle 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 $\pi_{i^*}^{s^*,\ell^*}$ choose $T_C = g^{t_C}$ for random $t_C \xleftarrow{\$} \mathbb{Z}_q$, $\mathcal{B}$ defines $T_C := g^u$. Similarly, the Diffie–Hellman share T_S of $\pi_{j^*}^{t^*,\ell_t^*}$ is defined as $T_S := g^v$. Finally, the master secret of $\pi_{i^*}^{s^*,\ell^*}$ is set equal to R.

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

Note that in any case algorithm $\mathcal{B}$ knows the master secret of $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$, and thus is able to compute all further protocol messages (in particular the finished messages fin_C and fin_S) and answer a potential Reveal-query to $\pi_{j^*}^{t^*,\ell_t^*}$ as required (note that there is no Reveal-query to $\pi_{i^*}^{s^*,\ell^*}$, 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 the above claim.

Game 6. In this game we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i^*}^{s^*}$ in phase ℓ^* with a random function. If $\pi_{j^*}^{t^*,\ell_t^*}$ uses the same master secret $\widetilde{ms}$ as $\pi_{i^*}^{s^*,\ell^*}$ (cf. Game 5), then the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{j^*}^{t^*,\ell_t^*}$ 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 breaking the security of the pseudorandom function PRF, thus

$$\Pr[\mathsf{break}_5^{(1)}] \leq \Pr[\mathsf{break}_6^{(1)}] + \epsilon_{\mathsf{prf}} \ .$$

Game 7. Finally we use that the full transcript of all messages sent and received is used to compute the `Finished` messages, and that `Finished` messages are computed by evaluating a truly random function that is only accessible to $\pi_{i^*}^{s^*,\ell^*}$ and (possibly) $\pi_{j^*}^{t^*,\ell_t^*}$ due to Game 6. This allows us to show that any adversary has probability at most $\frac{1}{2^\mu}$ of making oracle $\pi_{i^*}^{s^*,\ell^*}$ accept without having a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$.

Thus, this game proceeds exactly like the previous game, except that the challenger now aborts if oracle $\pi_{i^*}^{s^*,\ell^*}$ accepts without having a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$. Thus we have $\Pr[\mathsf{break}_7^{(1)}] = 0$.

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 $\pi_{i^*}^{s^*,\ell^*}$, the adversary receives no information about $F_{\widetilde{ms}}(label_3\|m_1\|\cdots\|m_{12})$. Therefore we have $\Pr[\mathsf{break}_7^{(1)}] = 2^{-\mu}$ and

$$\Pr[\mathsf{break}_6^{(1)}] \leq \Pr[\mathsf{break}_7^{(1)}] + \frac{1}{2^\mu} = \frac{2}{2^\mu} \ .$$

□

4.2.2 Proof of Lemma 1: $\epsilon_{\mathsf{server}}$

Proof. We now show that the probability that there exists an oracle with $\rho = \mathsf{Server}$ that accepts maliciously in the sense of Definition 4 is negligible. Let $\mathsf{break}_\delta^{(2)}$ be the event that occurs when the first oracle that accepts maliciously in the sense of Definition 4 with $\rho = \mathsf{Server}$ in Game δ.

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

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

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

$$\Pr[\mathsf{break}_0^{(2)}] \leq \Pr[\mathsf{break}_1^{(2)}] + \frac{(n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}})^2}{2^\lambda} \ .$$

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

$$\Pr[\mathsf{break}_1^{(2)}] = (n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}}) \cdot \Pr[\mathsf{break}_2^{(2)}] \ .$$

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

Game 3. The challenger proceeds as before, but we add an abort rule. We want to make sure that $\pi_{i^*}^{s^*,\ell^*}$ 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 $\pi_{i^*}^{s^*,\ell^*}$ 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 σ_C is a valid signature but there exists no oracle π_j^t which has previously output σ_C. Clearly we have

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

Note that the experiment is aborted if $\pi_{i^*}^{s^*,\ell^*}$ does not accept *maliciously*, due to Game 2. This means that party P_j must not be corrupted when $\pi_{i^*}^{s^*}$ accepts. To show that $\Pr[\mathsf{abort}_{\mathsf{sig}}] \leq (n_{\mathrm{pa}} n_{\mathrm{ke}}) \cdot \epsilon_{\mathsf{sig}}$, we construct a signature forger as follows. The forger receives as input a public key pk^* and simulates the challenger for $\mathcal{A}$. It guesses two indices $\phi_1 \xleftarrow{\$} [1, n_{\mathrm{pa}}]$ and $\phi_2 \xleftarrow{\$} [1, n_{\mathrm{ke}}]$, sets $pk_{\phi_1,\phi_2} = pk^*$, 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_{ϕ_1,ϕ_2} when necessary.

If $\phi_1 = j$ and the corresponding public key is pk_{j,ϕ_2}, which happens with probability $1/(n_{\mathrm{pa}} n_{\mathrm{ke}})$, then the forger can use the signature received by $\pi_{i^*}^{s^*,\ell^*}$ to break the EUF-CMA security of the signature scheme with success probability ϵ_{sig}, so $\Pr[\mathsf{abort}_{\mathsf{sig}}]/(n_{\mathrm{pa}} n_{\mathrm{ke}}) \leq \epsilon_{\mathsf{sig}}$. Therefore if $\Pr[\mathsf{abort}_{\mathsf{sig}}]$ is not negligible, then ϵ_{sig} is not negligible as well and we have

$$Pr[\mathsf{break}_2^{(2)}] \leq \Pr[\mathsf{break}_3^{(2)}] + n_{\mathrm{pa}} n_{\mathrm{ke}} \epsilon_{\mathsf{sig}} \quad .$$

Note that in Game 3 oracle $\pi_{i^*}^{s^*,\ell^*}$ receives as input a Diffie–Hellman value T_C such that T_C was chosen in some phase by another oracle, but not by the attacker. Note also that this phase of 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 and phase with $\pi_{j^*}^{t^*,\ell_t^*}$.

Note also that $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$, respectively.

Game 4. In this game, we replace the premaster secret $pms = g^{t_C t_S}$ shared by $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$, due to the modifications introduced in the previous games, provides us with the ability to prove indistinguishability under the Decisional Diffie–Hellman assumption.

Technically, the challenger in Game 4 proceeds as before, but when $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ 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 $\mathcal{B}$ solving the DDH problem as follows. Algorithm $\mathcal{B}$ 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$, respectively. Instead of computing the Diffie–Hellman key as in Game 3, it sets $pms = g^w$ both for the client and the server oracle. Now if $w = uv$, then this game proceeds exactly like Game 3, while if w is random then this game proceeds exactly like Game 4. Thus,

$$\Pr[\mathsf{break}_3^{(2)}] \leq \Pr[\mathsf{break}_4^{(2)}] + \epsilon_{\mathsf{ddh}} \quad .$$

Note that in Game 4 the premaster secret of $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ is uniformly random and independent of T_C and T_S. This will provide us with the ability 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ is chosen uniformly at random, independently 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 breaking the security of the pseudorandom function PRF, thus

$$\Pr[\mathsf{break}_4^{(2)}] \leq \Pr[\mathsf{break}_5^{(2)}] + \epsilon_{\mathsf{prf}} \quad .$$

Game 6. In this game we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ with a random function. Of course the same random function is used for both oracles $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$. 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 breaking the security of the pseudorandom function PRF, thus

$$\Pr[\mathsf{break}_5^{(2)}] \leq \Pr[\mathsf{break}_6^{(2)}] + \epsilon_{\mathsf{prf}} \quad .$$

Game 7. Finally we use that the full transcript of all messages sent and received is used to compute the `Finished` messages, and that `Finished` messages are computed by evaluating a truly random function that is only accessible to $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ due to Game 6. This allows us to show that any adversary has probability at most $1/2^\mu$ of making oracle $\pi_{i^*}^{s^*,\ell^*}$ accept without having a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$.

Thus, this game proceeds exactly like the previous game, except that the challenger now aborts if oracle $\pi_{i^*}^{s^*,\ell^*}$ accepts without having a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$. Therefore we have $\Pr[\mathsf{break}_7^{(1)}] = 0$.

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 $\pi_{i^*}^{s^*,\ell^*}$, the adversary receives no information about $F_{\widetilde{ms}}(label_3 \| m_1 \| \cdots \| m_{10})$. Thus we have

$$\Pr[\mathsf{break}_6^{(1)}] \leq \Pr[\mathsf{break}_7^{(1)}] + \frac{1}{2^\mu} = \frac{1}{2^\mu}.$$

□

Collecting probabilities of both previous sections yields Lemma 1. We obtain that

$$\begin{aligned}\epsilon_{\mathsf{auth}} &\leq \epsilon_{\mathsf{client}} + \epsilon_{\mathsf{server}} \\ &\leq 2\epsilon_{\mathsf{auth}} + n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + \epsilon_{\mathsf{sLHAE}}\right) \\ &\leq 4 n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\frac{n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}}}{2^\lambda} + n_{\mathrm{pa}} n_{\mathrm{ke}} \epsilon_{\mathsf{sig}} + \frac{5}{4}\epsilon_{\mathsf{ddh}} + \frac{5}{2}\epsilon_{\mathsf{prf}} + \frac{\epsilon_{\mathsf{sLHAE}}}{4} + n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^\mu}\right)\right) .\end{aligned}$$

4.2.3 Proof of confidentiality

Lemma 2. *For any adversary $\mathcal{A}$ running in time $\tau' \approx t$, the probability that $\mathcal{A}$ answers the encryption-challenge correctly is at most $1/2 + \epsilon_{\mathsf{enc}}$ with*

$$\epsilon_{\mathsf{enc}} \leq \epsilon_{\mathsf{auth}} + n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + \epsilon_{\mathsf{sLHAE}}\right) \ ,$$

where ϵ_{auth} is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 4 (cf. Lemma 1) and all other quantities are defined as stated in Theorem 1.

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

Game 0. This game equals the ACCE security experiment used in Definition 4. For some ϵ_{enc} we have

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

Game 1. The challenger in this game proceeds as before, but it aborts and chooses b' uniformly random if there exists any oracle that accepts maliciously in any phase in the sense of Definition 6. Thus we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{auth}} \ ,$$

where ϵ_{auth} is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 4 (cf. Lemma 1).

Recall that we assume that $\mathcal{A}$ always outputs (i, s, ℓ, b') such that all conditions in Property 2 of Definition 4 are satisfied. In particular it outputs (i, s, ℓ, b') such that $\pi_i^{s,\ell}$ accepts with intended partner $\Pi = j$, and P_j is not corrupted. Note that in Game 1 for any such phase $\pi_i^{s,\ell}$ there exists a unique partner phase π_j^{t,ℓ_t} such that $\pi_i^s.\mathtt{phases}[\ell].T$ has a matching conversation to $\pi_j^t.\mathtt{phases}[\ell_t].T$, as the game is aborted otherwise.

Game 2. The challenger in this game proceeds as before, but in addition guesses indices $(i^*, s^*, \ell^*) \xleftarrow{\$} [1, n_{\mathrm{pa}}] \times [1, n_{\mathrm{se}}] \times [1, n_{\mathrm{ph}}]$. It aborts and chooses b' at random if the attacker outputs (i, s, ℓ, b') with $(i, s, \ell) \neq (i^*, s^*, \ell^*)$. With probability $1/(n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}})$ we have $(i, s, \ell) = (i^*, s^*, \ell^*)$, and thus

$$\mathsf{Adv}_1 \leq n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \mathsf{Adv}_2 \ .$$

Note that in Game 2 we know that $\mathcal{A}$ will output (i^*, s^*, ℓ^*, b'). Note also that $\pi_{i^*}^{s^*,\ell^*}$ has a unique partner due to Game 1. In the sequel we denote with $\pi_{j^*}^{t^*,\ell_t^*}$ the unique oracle and phase such that $\pi_{i^*}^{s^*,\ell^*}$ has a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$, and say that $\pi_{j^*}^{t^*,\ell_t^*}$ is the *partner* of $\pi_{i^*}^{s^*,\ell^*}$.

Game 3. The challenger in this game proceeds as before, but replaces the premaster secret pms of $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ 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 $\pi_{j^*}^{t^*,\ell_t^*}$, respectively, as otherwise $\pi_{i^*}^{s^*,\ell^*}$ would not have a matching conversation to $\pi_{j^*}^{t^*,\ell_t^*}$ and the game would be aborted. Thus, both oracles compute the premaster secret as $pms = g^{uv}$. Let $T_{i^*,s^*,\ell^*} = g^u$ denote the Diffie–Hellman share chosen by $\pi_{i^*}^{s^*}$ in phase ℓ, and let $T_{j^*,t^*,\ell_t^*} = g^v$ denote the share chosen by its partner $\pi_{j^*}^{t^*}$ in phase ℓ_t.

Suppose that there exists an algorithm $\mathcal{A}$ distinguishing Game 3 from Game 2. Then we can construct an algorithm $\mathcal{B}$ solving the DDH problem as follows. $\mathcal{B}$ receives as input (g, g^u, g^v, g^w). It implements the challenger for $\mathcal{A}$ as in Game 2, except that it sets $T_{i^*,s^*,\ell^*} := g^u$ and $T_{j^*,t^*,\ell_t^*} := g^v$, and the premaster secret of $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ equal to $pms := g^w$. Note that $\mathcal{B}$ can simulate all messages exchanged between $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ properly, in particular the finished messages using knowledge of $pms = g^w$. Since all other oracles are not modified, $\mathcal{B}$ can simulate these oracles properly as well.

If $w = uv$, then the view of $\mathcal{A}$ when interacting with $\mathcal{B}$ is identical to Game 2, while if $w \xleftarrow{\$} \mathbb{Z}_q$ then it is identical to Game 3. Thus,

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{ddh}} \ .$$

Game 4. In Game 4 we make use of the fact that the premaster secret $\widetilde{pms}$ of $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ 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 4 from Game 3 implies an algorithm breaking the security of the pseudorandom function PRF, thus

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

Game 5. In this game we replace the function $\mathsf{PRF}(\widetilde{ms}, \cdot)$ used by $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ with a random function $F_{\widetilde{ms}}$. Of course the same random function is used for both oracles (in their respective phases) $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$. 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 5 from Game 4 again implies an algorithm breaking the security of the pseudorandom function PRF. Moreover, in Game 5 the adversary always receives a random key in response to a Test query, and thus receives no information about b', which implies $\mathsf{Adv}_5 = 0$ and

$$\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{prf}} = \epsilon_{\mathsf{prf}} \ .$$

Note that in Game 5 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ is uniformly random and independent of all TLS handshake messages exchanged in the pre-accept phase.

Game 6. 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 $\pi_{i^*}^{s^*,\ell^*}$ and $\pi_{j^*}^{t^*,\ell_t^*}$ in the stateful symmetric encryption scheme uniformly at random and independent of all TLS handshake messages.

In this game we construct a simulator $\mathcal{B}$ that uses a successful ACCE attacker $\mathcal{A}$ to break the security of the underlying sLHAE secure symmetric encryption scheme. By assumption, the simulator $\mathcal{B}$ is given access to an encryption oracle Encrypt and a decryption oracle Decrypt. $\mathcal{B}$ embeds the sLHAE experiment by simply forwarding all $\mathsf{Encrypt}(\pi_{i^*}^{s^*,\ell^*},\cdot)$ queries to Encrypt, and all $\mathsf{Decrypt}(\pi_{j^*}^{t^*,\ell_t^*},\cdot)$ queries to Decrypt. Otherwise it proceeds as the challenger in Game 5.

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

$$\mathsf{Adv}_5 = \mathsf{Adv}_6 \ .$$

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

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

$$\mathsf{Adv}_6 \leq 1/2 + \epsilon_{\mathsf{sLHAE}} \ .$$

□

Addig up probabilities from Lemmas 1 and 2, we obtain that

$$\begin{aligned}\epsilon_{\mathsf{tls}} &\leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{enc}} \\ &\leq 2\epsilon_{\mathsf{auth}} + n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\epsilon_{\mathsf{ddh}} + 2\epsilon_{\mathsf{prf}} + \epsilon_{\mathsf{sLHAE}}\right) \\ &\leq 4 n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(n_{\mathrm{pa}} n_{\mathrm{ke}} \epsilon_{\mathsf{sig}} + \frac{5}{4}\epsilon_{\mathsf{ddh}} + \frac{5}{2}\epsilon_{\mathsf{prf}} + \frac{1}{4}\epsilon_{\mathsf{sLHAE}} + n_{\mathrm{pa}} n_{\mathrm{se}} n_{\mathrm{ph}} \left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{prfodh}} + \epsilon_{\mathsf{prf}} + \frac{1}{2^{\mu}} \right) \right)\end{aligned}$$

which yields Theorem 1.

Note, that we do lose some tightness compared to the original ACCE proof of `TLS_DHE_DSS_`: for the authentication game, we additionally have to guess the phase in which the adversary makes an oracle maliciously accept, and for the encryption game we also have to guess the phase to which we input the challenge keys.

4.3 Multi-phase secure TLS with SCSV/RIE is a weakly secure renegotiable ACCE

Theorem 2 (TLS with SCSV/RIE is a weakly secure renegotiable ACCE)**.** *Let Π be a TLS ciphersuite without countermeasures, and let $R(\Pi)$ denote the TLS ciphersuite with SCSV/RIE countermeasures, as transformed according to Figure 5. If $R(\Pi)$ is a $(\tau, \epsilon_{\mathsf{mp}})$-secure multi-phase ACCE protocol, and* PRF *is a $(\tau, \epsilon_{\mathsf{prf}})$-secure pseudorandom function, then $R(\Pi)$ is a (τ, ϵ)-weakly secure renegotiable ACCE, with $\epsilon = \epsilon_{\mathsf{mp}} + \epsilon_{\mathsf{prf}}$.*

Intuitively, the use of the RIE countermeasure guarantees that each party who renegotiates has the same view of (a) whether they are renegotiating, and (b) which handshake is the 'previous' handshake. We can chain these together to obtain the requirement for a secure renegotiable ACCE: parties who renegotiate have the same view of all previous handshakes. If this is violated, either the non-renegotiable aspects of TLS have been broken, or a collision has been found in the computation of the renegotiation indication extension.

Proof. Suppose $\mathcal{A}$ breaks the weak renegotiable ACCE security of the protocol $R(\Pi)$. We will show that either $\mathcal{A}$ directly breaks the multi-phase ACCE security of $R(\Pi)$ or $\mathcal{A}$ can be used to construct another algorithm that breaks either the security of the PRF or the multi-phase ACCE security of $R(\Pi)$.

We approach the proof in three cases: either $\mathcal{A}$ has broken the confidentiality/integrity of the weakly secure renegotiable ACCE, or $\mathcal{A}$ has broken the weak renegotiation authentication of the weakly secure renegotiable ACCE, and the latter can happen by meeting either condition $\mathbf{M'(a)}$ or $\mathbf{M'(b)}$.

Confidentiality/integrity. Since the winning conditions for the confidentiality/integrity part of the security game are the same for both definitions, every adversary who breaks confidentiality/integrity in the weakly secure renegotiable ACCE security game for $R(\Pi)$ directly breaks confidentiality/integrity in the multi-phase ACCE security game for $R(\Pi)$.

Authentication — $\mathbf{M'(a)}$. Suppose $\mathcal{A}$ wins the weak renegotiable ACCE security experiment for $R(\Pi)$ using condition $\mathbf{M'(a)}$. Either there is no ℓ at all such that $\pi_B^t.\mathtt{phases}[\ell].T$ matches $\pi_A^s.\mathtt{phases}[\ell^*].T$, or there is such an ℓ but $\ell \neq \ell^*$.

First consider the case where there is no ℓ at all such that $\pi_B^t.\mathtt{phases}[\ell].T$ matches $\pi_A^s.\mathtt{phases}[\ell^*].T$. That meets condition $\mathbf{M}$ of Definition 4 for $R(\Pi)$.

Now consider the case where there is an ℓ such that $\pi_B^t.\mathtt{phases}[\ell].T$ matches $\pi_A^s.\mathtt{phases}[\ell^*].T$ but $\ell \neq \ell^*$. Assume without loss of generality $\ell < \ell^*$ (otherwise we could swap the oracles).

There must exist some value $j \in [1, \ell - 1]$ such that $\pi_A^s.\mathtt{phases}[\ell^* - j].T \neq \pi_B^t.\mathtt{phases}[\ell - j].T$. In particular, at worst $j = \ell - 1$, since in π_B^t's first phase its outgoing message m_1 contains $ext_C = \mathtt{empty}$ but π_A^s received a message m_1 with $ext_c \neq \mathtt{empty}$. Let j be minimal. Then $\pi_B^t.\mathtt{phases}[\ell - j + 1].T$ matches $\pi_A^s.\mathtt{phases}[\ell^* - j + 1].T$. In particular, messages m_1 of those two transcripts are equal, and so are messages m_2 of those two transcripts. Since RIE is being used, m_1 and m_2 contain $fin_C^{(-1)}$ and $fin_S^{(-1)}$, and since π_A^{s,ℓ^*-j+1} accepted, both π_A^{s,ℓ^*-j+1} and $\pi_B^{t,\ell-j+1}$ used the same $fin_C^{(-1)}$ and $fin_S^{(-1)}$ values. But at each party, $fin_C^{(-1)}$ and $fin_S^{(-1)}$ are their hash (using a PRF) of the handshake transcripts from phases π_A^{s,ℓ^*-j} and $\pi_B^{t,\ell-j}$, and we know that these handshake transcripts are not equal. This means a collision has occurred in PRF, which happens with negligible probability.

Thus, assuming PRF is secure and $R(\Pi)$ is a secure multi-phase ACCE, no $\mathcal{A}$ can achieve conditions $\mathbf{M'(a)}$ and $\mathbf{A1}$–$\mathbf{A5}$.

Authentication — $\mathbf{M'(b)}$. Now suppose $\mathcal{A}$ wins the weak renegotiable ACCE security experiment for $R(\Pi)$ using condition $\mathbf{M'(b)}$ but not $\mathbf{M'(a)}$. In particular, for every $\ell' < \ell^*$, $\pi_A^s.\mathtt{phases}[\ell'].T = \pi_B^t.\mathtt{phases}[\ell'].T$ *but* there is some $\ell < \ell^*$ such that $\pi_A^s.\mathtt{phases}[\ell].RT \neq \pi_B^t.\mathtt{phases}[\ell].RT$. Choose ℓ minimal. Let v be the smallest index such that the vth ciphertext C_v of $\pi_A^s.\mathtt{phases}[\ell].RT$ is not equal to the vth ciphertext of $\pi_B^t.\mathtt{phases}[\ell].RT$.

Assume without loss of generality that C_v was received by π_A^s as the vth ciphertext but was not sent by π_B^t as the vth ciphertext. (The alternative is that C_v was *sent* by π_A^s as the vth ciphertext but was not received by π_B^t as the vth ciphertext. However, we could then focus on everything from π_B^t's perspective and apply the same argument.)

This means that when $\mathcal{A}$ called $\mathsf{Decrypt}(\pi_A^s, C_v, head)$, if $b = 0$ then $\mathsf{Decrypt}$ returned $(\perp, \cdot)$, whereas if $b = 1$ then $\mathsf{Decrypt}$ returned $(m', \cdot)$ where $m' \neq \perp$. Our simulator can thus output (A, s, ℓ, b') for its guess of b' as above, and this will equal b with probability at least ϵ, making condition $\mathbf{C6}$ hold in Definition 4. We need to show that conditions $\mathbf{C1}$–$\mathbf{C5}$ also hold for (A, s, ℓ).

Since $\mathcal{A}$ wins the weak renegotiable ACCE experiment using condition $\mathbf{M'(b)}$, we have that $\mathbf{A1}$–$\mathbf{A5}$ all hold. We want to show that, at the time that π_A^s accepted in phase $\ell + 1$, conditions $\mathbf{C1}$–$\mathbf{C5}$ also hold for (A, s, ℓ).

- $\mathbf{C1}$: $\mathbf{A1}$ directly implies $\mathbf{C1}$, since if π_A^s has rejected in any phase prior to ℓ^* then it would not have a phase ℓ^*.
- $\mathbf{C2}$ and $\mathbf{C3}$: Conditions $\mathbf{A2}$ and $\mathbf{A3}$ of Definition 6 do not imply that $\mathcal{A}$ did not ask $\mathsf{Corrupt}$ queries prohibited by $\mathbf{C2}$ and $\mathbf{C3}$. However, we do have that $\pi_A^s.\mathtt{phases}[\ell].T = \pi_B^t.\mathtt{phases}[\ell].T$; in other words, $\mathcal{A}$ was not *active* in the handshake for phase ℓ. Thus, $\mathcal{A}$ is *equivalent* to an adversary who did not ask any $\mathsf{Corrupt}$ queries for public keys used in phase ℓ until after π_A^s accepts in phase ℓ.
- $\mathbf{C4}$: $\mathbf{A4}$ directly implies $\mathbf{C4}$, at the time that π_A^s accepted.
- $\mathbf{C5}$: Since π_A^s chooses nonce r_C (if a client) or r_S (if a server) randomly, except with negligible probability there is no $\ell' < \ell$ such that $\pi_A^s.\mathtt{phases}[\ell'].T = \pi_A^s.\mathtt{phases}[\ell].T$. By $\mathbf{A5}$, $\mathcal{A}$ did not issue $\mathsf{Reveal}(\pi_B^t, \ell)$ before π_A^s accepted in phase $\ell + 1$. Thus at the time that π_A^s accepted, $\mathcal{A}$ did not issue $\mathsf{Reveal}(\pi_B^t, \ell')$ to any phase with $\pi_B^t.\mathtt{phases}[\ell'].T = \pi_A^s.\mathtt{phases}[\ell].T$, satisfying condition $\mathbf{C5}$.

Thus, assuming $R(\Pi)$ is a secure multi-phase ACCE no $\mathcal{A}$ can achieve conditions $\mathbf{M'(b)}$ and $\mathbf{A1}$–$\mathbf{A5}$. □

We can combine Theorems 1 and 2 to obtain:

Corollary 1 (TLS_DHE_DSS_ with SCSV/RIE is a weakly secure renegotiable ACCE)**.** *Under the same assumptions on the building blocks as in Theorem 1,* TLS_DHE_DSS_ *with SCSV/RIE countermeasures is a weakly secure renegotiable ACCE protocol.*

5 Renegotiation Security of TLS with a New Countermeasure

We now present a new TLS renegotiation countermeasure that provides integrity protection for the record layer transcript upon renegotiation (even when previous phases' session keys are leaked while the phase is still active), thereby achieving the full security of Definition 5. This countermeasure is quite straightforward: by including a hash of all record layer messages in the renegotiation information extension, parties can confirm that they share the same view of their previous record layers.

The renegotiation information extension already contains a fingerprint of the previous phrase's handshake transcript via the `client_verify_data` ($fin_C^{(-1)}$) and `server_verify_data` ($fin_S^{(-1)}$) values. We modify the renegotiation information extension to include an additional value, the fingerprint of the encrypted messages sent over the previous phase's record layer. In particular, if negotiating:

$$ext_C \leftarrow fin_C^{(-1)} \parallel \mathsf{PRF}(ms^{(-1)}, label_5 \| H(RT^{(-1)})) \ , \tag{1}$$

where $ms^{(-1)}$ is the previous phase's master secret, H is a collision-resistant hash function, and $RT^{(-1)}$ is the party's view of the previous phase's record layer transcript. Appropriate checks are performed by the server. With this additional information, the two parties will now not complete renegotiation unless they have matching views of the record layer transcripts from the previous phase.

Remark 4. Note that the proof does not require the server to also send its view of the record layer transcript; the server simply checks what it receives from the client and stops if it is not what it expects. The same is actually true as well of the RIE countermeasure, and the proof of Theorem 2 would go through if only ext_C contained $fin_S^{(-1)}$. However, if the security model is altered to allow $\mathsf{Corrupt}$s of the current phase's public keys but not Reveals of the previous phase's session keys, then having both ext_C and ext_S include each party's view of the the transcript is required to achieve security.

In practice, it is not difficult to, on an incremental basis, compute hashes of the ciphertexts sent and received over the record layer in that phase. In particular, it is not necessary to store all record layer messages to input to the hash function all at once, as common programming APIs for hash functions allow the hash value to be provided incrementally.[2] We note that there may be additional implementation details to consider, for example the need to distinguish the original renegotiation information extension from this augmented renegotiation information extension using a different `extension_type` value.

Theorem 3 (TLS with new countermeasure is a secure renegotiable ACCE)**.** *Let* Π *be a generic TLS ciphersuite without countermeasures, and let* $R'(\Pi)$ *denote the TLS ciphersuite with original RIE countermeasures as in Figure 5 but using* ext_C *as in equation (1). If* $R'(\Pi)$ *is a* $(\tau, \epsilon_{\mathsf{mp}})$*-secure multi-phase ACCE protocol,* H *is a* $(\tau, \epsilon_{\mathsf{h}})$*-collision-resistant hash function, and* PRF *is a* $(\tau, \epsilon_{\mathsf{prf}})$*-secure pseudorandom function, then* $R'(\Pi)$ *is a* (τ, ϵ)*-secure renegotiable ACCE, where* $\epsilon = \epsilon_{\mathsf{mp}} + \epsilon_{\mathsf{h}} + \epsilon_{\mathsf{prf}}$*.*

The proof proceeds similarly to that of Theorem 2. The main difference is that, in one case, the removal of restrictions **A4** and **A5** means we can no longer reduce down to a violation of confidentiality/integrity in the multi-phase security of $R'(\Pi)$, and instead have to rely on the new countermeasure to detect non-matching record layer transcripts and reduce to the security of the PRF and hash function.

Proof. The proof proceeds similarly to that of Theorem 2. Suppose $\mathcal{A}$ breaks the renegotiable ACCE security of the protocol $R'(\Pi)$. We will show that either $\mathcal{A}$ directly breaks the multi-phase ACCE security of $R'(\Pi)$ or $\mathcal{A}$ can be used to construct another algorithm that breaks either the security of the PRF, the collision resistance of H, or the multi-phase ACCE security of $R'(\Pi)$.

[2]See for example OpenSSL's API for SHA-1: a hash context is initialized with `SHA1_Init`, then the value to be hashed is provided in chunks by multiple calls to `SHA1_Update`, then the final hash output is computed by `SHA1_Final`. http://www.openssl.org/docs/crypto/sha.html

As before, we approach the proof in three cases: either $\mathcal{A}$ has broken the confidentiality/integrity of the secure renegotiable ACCE, or $\mathcal{A}$ has broken the renegotiation authentication of the secure renegotiable ACCE, and the latter can happen by meeting either condition $\mathbf{M'(a)}$ or $\mathbf{M'(b)}$. The first two cases, confidentiality and authentication for $\mathbf{M'(a)}$, proceed exactly as in the proof of Theorem 2.

Authentication — $\mathbf{M'(b)}$. Suppose $\mathcal{A}$ wins the renegotiable ACCE security experiment for $R'(\Pi)$ using condition $\mathbf{M'(b)}$ but not $\mathbf{M'(a)}$.

When conditions **A1**–**A5** hold, then the same argument as in the proof for case $\mathbf{M'(b)}$ of Theorem 2 still holds, in which case conditions **C1**–**C6** hold and $\mathcal{A}$ can be used to break the confidentiality/integrity of $R'(\Pi)$ in the multi-phase ACCE experiment.

However, in the secure renegotiable ACCE experiment, the adversary is no longer constrained by conditions **A4** and **A5**, so it can make Reveal queries while the phase is active. This means that conditions **C4** and **C5** are no longer satisfied, so we cannot reduce to the confidentiality/integrity of $R'(\Pi)$ in the multi-phase ACCE experiment when such Reveal queries are issued. Instead, we make use of the new countermeasure, and apply an argument similar to that for case $\mathbf{M'(a)}$ of Theorem 2.

If $\mathcal{A}$ wins using condition $\mathbf{M'(b)}$ but not $\mathbf{M'(a)}$, then, for every $\ell' < \ell^*$, $\pi_A^s.\mathtt{phases}[\ell'].T = \pi_B^t.\mathtt{phases}[\ell'].T$. But there is some $\ell < \ell^*$ such that $\pi_A^s.\mathtt{phases}[\ell].RT \neq \pi_B^t.\mathtt{phases}[\ell].RT$. Choose ℓ *maximal*. Then $\pi_B^t.\mathtt{phases}[\ell+1].T$ matches $\pi_A^s.\mathtt{phases}[\ell+1].T$. In particular, messages m_1 of those two transcripts are equal, and so are messages m_2 of those two transcripts. Since the new countermeasure is being used, m_2 contains ext_C, which contains $\mathsf{PRF}(ms^{(-1)}, label_5 \| H(RT^{(-1)}))$. Since $\pi_A^{s,\ell+1}$ accepted, both $\pi_A^{s,\ell+1}$ and $\pi_B^{t,\ell+1}$ used the same value in ext_C. But at each party, this value is the value of the PRF applied to the hash of the record layer transcripts from phases $\pi_A^{s,\ell}$ and $\pi_B^{t,\ell}$, which we know are not equal. This means a collision has occurred either in H or PRF, which happens with negligible probability.

Thus, assuming PRF is secure, H is collision-resistant, and $R'(\Pi)$ is a secure multi-phase ACCE, no $\mathcal{A}$ can achieve conditions $\mathbf{M'(a)}$ and **A1**–**A3**. □

A straightforward adaptation of the proof of Theorem 1 can be used to show that `TLS_DHE_DSS_` with the new countermeasure is a secure multi-phase ACCE, since due to the security of the PRF and the master secret an adversary cannot learn any information about the record layer transcript, even given the new ext_C value. Combining that observation with Theorem 3:

Corollary 2 (`TLS_DHE_DSS_` with new countermeasure is a secure renegotiable ACCE)**.** *Under the same assumptions on the building blocks as in Theorem 1 and that H is a collision-resistant hash function,* `TLS_DHE_DSS_` *with the new countermeasure is a secure renegotiable ACCE protocol.*

6 Conclusion

Although two-party protocols for establishing secure communication have been extensively studied in the literature and are widely used in practice, this is the first work to consider the important issue of renegotiation, in which parties update one or more aspects of their connection — their authentication credentials, cryptographic parameters, or simply refresh their session key. The importance of correctly implementing renegotiation was highlighted by the 2009 attack of Ray and Dispensa on how certain applications process data from renegotiable TLS connections.

We have developed a formal model for describing the security of renegotiable cryptographic protocols, focussing on authenticated and confidential channel establishment (ACCE) protocols. We have specifically analyzed renegotiation in the TLS protocol, identifying the original attack of Ray and Dispensa in our model, and then analyzing the security of the SCSV/RIE countermeasure as well as a new countermeasure that provides security in the face of stronger adversaries.

Renegotiation, reauthentication, and rekeying are important features of many other applied cryptographic protocols. Future applied work includes examining the security of rekeying in protocols such as SSH or IKEv2 in our model. Open theoretical questions include how to adapt our approach for defining secure renegotiation to other primitives, in particular authenticated key exchange protocols. The overall security of TLS still has many important open questions, including the security of other TLS ciphersuites, one-way authentication, and the analysis of security when renegotiation from one ciphersuite to another. Given that attacks continue to be found outside the core key agreement component of TLS, further research into modelling the security of TLS in increasingly realistic scenarios is well-motivated.

Acknowledgements

The authors gratefully acknowledge helpful discussions with Colin Boyd, Jörg Schwenk, and Cas Cremers.

References

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, *Lecture Notes in Computer Science*, volume 2020, pp. 143–158. Springer, April 2001. DOI:10.1007/3-540-45353-9_12.

[BFS+12] Christina Brzuska, Mark Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: Relaxed yet composable security notions for key exchange, 2012. EPRINT `http://eprint.iacr.org/2012/242`. Cryptology ePrint Archive, Report 2012/242.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, *Lecture Notes in Computer Science*, volume 773, pp. 232–249. Springer, August 1994. DOI:10.1007/3-540-48329-2_21.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, *Lecture Notes in Computer Science*, volume 4004, pp. 409–426. Springer, May / June 2006. DOI:10.1007/11761679_25.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, *Lecture Notes in Computer Science*, volume 1355, pp. 30–45. Springer, December 1997. DOI:10.1007/BFb0024447.

[BWNH+03] Simon Blake-Wilson, Magnus Nystroem, David Hopwood, Jan Mikkelsen, and Tim Wright. Transport Layer Security (TLS) extensions, June 2003. URL `http://www.ietf.org/rfc/rfc3546.txt`. RFC 3546.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, *Lecture Notes in Computer Science*, volume 2045, pp. 453–474. Springer, May 2001. DOI:10.1007/3-540-44987-6_28.

[CK02] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, *Lecture Notes in Computer Science*, volume 2442, pp. 143–161. Springer, August 2002. DOI:10.1007/3-540-46035-7_22. `http://eprint.iacr.org/2002/120/`.

[DA99] Tim Dierks and Christopher Allen. The TLS protocol version 1.0, January 1999. URL `http://www.ietf.org/rfc/rfc2246.txt`. RFC 2246.

[DR06] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) protocol version 1.1, April 2006. URL `http://www.ietf.org/rfc/rfc4346.txt`. RFC 4346.

[DR08] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) protocol version 1.2, August 2008. URL `http://www.ietf.org/rfc/rfc5246.txt`. RFC 5246.

[Far10] Stephen Farrell. Why didn't we spot that? *IEEE Internet Computing*, **14**(1):84–87, Jan.–Feb. 2010. DOI:10.1109/MIC.2010.21.

[FKK11] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The Secure Sockets Layer (SSL) protocol version 3.0, August 2011. URL `http://www.ietf.org/rfc/rfc6101.txt`. RFC 6101; republication of original SSL 3.0 specification by Netscape of November 18, 1996.

[Gel12] Rati Gelashvili. Attacks on re-keying and renegotiation in key exchange protocols, April 2012. Bachelor's thesis, ETH Zurich.

[GMP+08] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally composable security analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Second International Conference on Provable Security (ProvSec) 2008*, *Lecture Notes in Computer Science*, volume 5324, pp. 313–327. Springer, 2008. DOI:10.1007/978-3-540-88733-1_22. Full version available as `http://eprint.iacr.org/2008/251`.

[JKJ02] Jakob Jonsson and Burton S. Kaliski Jr. On the security of RSA encryption in TLS. In Moti Yung, editor, *Advances in Cryptology – Proc. CRYPTO 2002*, *Lecture Notes in Computer Science*, volume 2442, pp. 183–199. Springer, 2002. DOI:10.1007/3-540-45708-9_9.

[JKSS11] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model (full version). Cryptology ePrint Archive, Report 2011/219, 2011. http://eprint.iacr.org/2011/219.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, *Lecture Notes in Computer Science*, volume 7417, pp. 273–293. Springer, August 2012. DOI:10.1007/978-3-642-32009-5_17.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, *Lecture Notes in Computer Science*, volume 2139, pp. 310–331. Springer, August 2001. DOI:10.1007/3-540-44647-8_19.

[Lan11] Adam Langley. Google online security blog: Protecting data for the long term with forward secrecy, November 2011. http://googleonlinesecurity.blogspot.com/2011/11/protecting-data-for-long-term-with.html.

[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, *Lecture Notes in Computer Science*, volume 4784, pp. 1–16. Springer, November 2007. DOI:10.1007/978-3-540-75670-5_1.

[MSW08] Paul Morrissey, Nigel P. Smart, and B. Warinschi. A modular security analysis of the TLS handshake protocol. In Josef Pieprzyk, editor, *Advances in Cryptology – Proc. ASIACRYPT 2008*, *Lecture Notes in Computer Science*, volume 5350, pp. 55–73. Springer, 2008. DOI:10.1007/978-3-540-89255-7_5. Full version available as http://eprint.iacr.org/2008/236.

[PRS11] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, *Lecture Notes in Computer Science*, volume 7073, pp. 372–389. Springer, December 2011. DOI:10.1007/978-3-642-25385-0_20.

[RD09] Marsh Ray and Steve Dispensa. Renegotiating TLS, November 2009. http://extendedsubset.com/Renegotiating_TLS.pdf.

[Res00] Eric Rescorla. HTTP over TLS, May 2000. URL http://www.ietf.org/rfc/rfc2818.txt. RFC 2818.

[RRDO10] Eric Rescorla, Marsh Ray, Steve Dispensa, and Nasko Oskov. Transport Layer Security (TLS) renegotiation indication extension, February 2010. URL http://www.ietf.org/rfc/rfc5746.txt. RFC 5746.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, Nov 2004. URL http://eprint.iacr.org/2004/332.

[Zol09] Thierry Zoller. TLS & SSLv3 renegotiation vulnerability. Technical report, G-SEC, 2009. http://www.g-sec.lu/practicaltls.pdf.

A Additional Definitions

A.1 Stateful Length-Hiding Authenticated Encryption (sLHAE)

Length-hiding authenticated encryption (LHAE) was originally introduced by Paterson *et al.* [PRS11]. Here we describe the variant stateful LHAE (sLHAE), stated by Jager *et al.* [JKSS12].

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

Definition 7. *A stateful symmetric encryption scheme* $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$ *is* $(\tau, \epsilon_{\mathsf{sLHAE}})$*-secure, if* $\Pr[b = b'] \le \epsilon_{\mathsf{sLHAE}}$ *for all adversaries* $\mathcal{A}$ *running in time at most* τ *in the following experiment:*

1. *Choose* $b \xleftarrow{\$} \{0,1\}$ *and* $k \xleftarrow{\$} \{0,1\}^\kappa$*, and set* $st_e \leftarrow \emptyset$ *and* $st_d \leftarrow \emptyset$*.*
2. *Run* $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}$.

Oracle $\mathsf{Encrypt}(m_0, m_1, len, head)$ *takes as input two messages* m_0 *and* m_1 *and a length* len*, and keeps a counter* i *which is initialized to* 0*. Oracle* $\mathsf{Decrypt}(C, head)$ *takes as input a ciphertext* C *and header* $head$*, and keeps a counter* j *and a switch* diverge*, both initialized to* 0*. Both oracles proceed as defined in Figure 3.*

This behaviour of the Decrypt oracle in this combined definition for confidentiality and integrity can be somewhat difficult to understand, so we give a brief explanation. From a high level, when $b = 0$, the adversary always receives $\perp$ from Decrypt queries. When $b = 1$, if the adversary successfully manipulates the integrity of the scheme — either by injecting a new valid ciphertext or messing around with the order of ciphertexts but still getting them to successfully decrypt — then Decrypt outputs the message m which is not equal to $\perp$, so the adversary learns that $b = 1$ not 0.

Note that diverge is stateful, so if it is ever set to 1 it remains 1. This allows the experiment to capture the following property. Suppose the adversary calls Encrypt to get a (valid) ciphertext C_1, then calls Decrypt on an invalid ciphertext C'_1. Next, the adversary calls Encrypt to get another (valid) ciphertext C_2, then calls Decrypt on C_2. Since the adversary previously called Decrypt on an invalid ciphertext, it should be that st_d has 'diverged' and all future ciphertexts should *not* decrypt: thus if C_2 does decrypt successfully, the scheme should be considered insecure.

```
Encrypt($m_0, m_1, len, head$):
1. $i \leftarrow i + 1$
2. $(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, len, head, m_0, st_e)$
3. $(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, len, head, m_1, st_e)$
4. If $(C^{(0)} = \perp)$ OR $(C^{(1)} = \perp)$, then return $\perp$
5. $(C_i, st_e) \leftarrow (C^{(b)}, st_e^{(b)})$
6. Return $C_i$

Decrypt($C, head$):
1. $j \leftarrow j + 1$
2. $(m, st_d) = \mathsf{StE.Dec}(k, head, C, st_d)$
3. If $(j > i)$ OR $(C \neq C_j)$, then diverge $\leftarrow 1$
4. If $(b = 1)$ AND (diverge $= 1$), then return $m$
5. Else return $\perp$
```

Figure 3: Encrypt and Decrypt oracles in the stateful LHAE security experiment.

A.2 The PRF-Oracle-Diffie-Hellman Assumption

The PRF-Oracle-Diffie-Hellman (PRFODH) assumption presented by Jager *et al.* [JKSS12] is a variant of the ODH assumption introduced by Abdalla, Bellare and Rogaway [ABR01], adapted from hash functions to PRFs. Jager *et al.* allow a single oracle query, in contrast to a polynomial number of queries as in the original assumption [ABR01]. Abdalla *et al.* point out that the ODH assumption is heuristically reasonable: in the random oracle model, the strong DH assumption implies the ODH assumption.

Let G be a group with generator g. Let PRF be a deterministic function $\mathsf{PRF} : G \times \{0,1\}^* \rightarrow \{0,1\}^\mu$. Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

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

Definition 8. *We say that the* PRFODH *problem is* $(\tau, \epsilon_{\mathsf{prfodh}})$*-hard with respect to* G *and* PRF*, if for all adversaries* $\mathcal{A}$ *that run in time* τ *it holds that* $|\Pr[b = b'] - 1/2| \leq \epsilon_{\mathsf{prfodh}}$.

B Protocols without forward security

B.1 Model

The ACCE notion of Jager *et al.* [JKSS12] requires confidentiality/integrity of ciphertexts to hold even when the long-term secret keys are corrupted after the handshake completes. As a result, RSA key transport-based ciphersuites cannot be proven secure in this model: they do not have forward security, since corrupting the long-term secret keys allows the adversary to compute the session key.

It is plausible that a non-forward-secure notion of ACCE could be defined in which an RSA key transport-based ciphersuite could be proven secure. While that is beyond the scope of this document, we do comment on how the notions we have introduced in this paper may be modified to consider protocols without forward security.

For renegotiation authentication, public keys used in phase ℓ cannot be $\mathsf{Corrupt}$ed while the post-accept stage of phase ℓ is still active, as it could allow the adversary to compute the session key and thus inject messages undetectably. One could add the following restrictions to Def. 6 to consider protocols without forward security:

A6. $\mathcal{A}$ did not query $\mathsf{Corrupt}(P_A, \pi_A^s.\mathtt{phases}[\ell].pk)$ before π_A^s accepted in phase $\ell+1$, for every $\ell < \ell^*$; and

A7. $\mathcal{A}$ did not query $\mathsf{Corrupt}(P_B, \pi_A^s.\mathtt{phases}[\ell].pk')$ before π_A^s accepted in phase $\ell+1$, for every $\ell < \ell^*$, where $\pi_A^s.d = \pi_B^t$.

For confidentiality/integrity, public keys used in phase ℓ can never be $\mathsf{Corrupt}$ed, as it could allow the adversary to compute the session key and distinguish ciphertexts. One could replace the following restrictions in Def. 3 to consider protocols without forward security:

C2′. $\mathcal{A}$ did not query $\mathsf{Corrupt}(P_A, \pi_A^s.\mathtt{phases}[\ell].pk)$; and

C3′. $\mathcal{A}$ did not query $\mathsf{Corrupt}(P_B, \pi_A^s.\mathtt{phases}[\ell].pk')$.

B.2 On renegotiation security of `TLS_RSA_` with SCSV/RIE

As mentioned in Section 2, TLS ciphersuites without forward security, such as RSA key transport, cannot be proven to be secure ACCE protocols, since the compromise of a party's long-term secret key after the phase accepts allows the adversary to compute the master secret. Hence RSA key transport-based ciphersuites can also not be shown to be secure or weakly secure renegotiable ACCE protocols. However, it is plausible that the ACCE definition can be modified to consider protocols without forward security, in which case RSA key transport may be able to proven secure. In such a scenario, it should be possible to show that such ciphersuites, when using SCSV/RIE, also satisfy a non-forward-secure notion of weak renegotiation security; or when using the new countermeasure in Section 5, also satisfy a non-forward-secure notion of renegotiation security.

C TLS Session Resumption

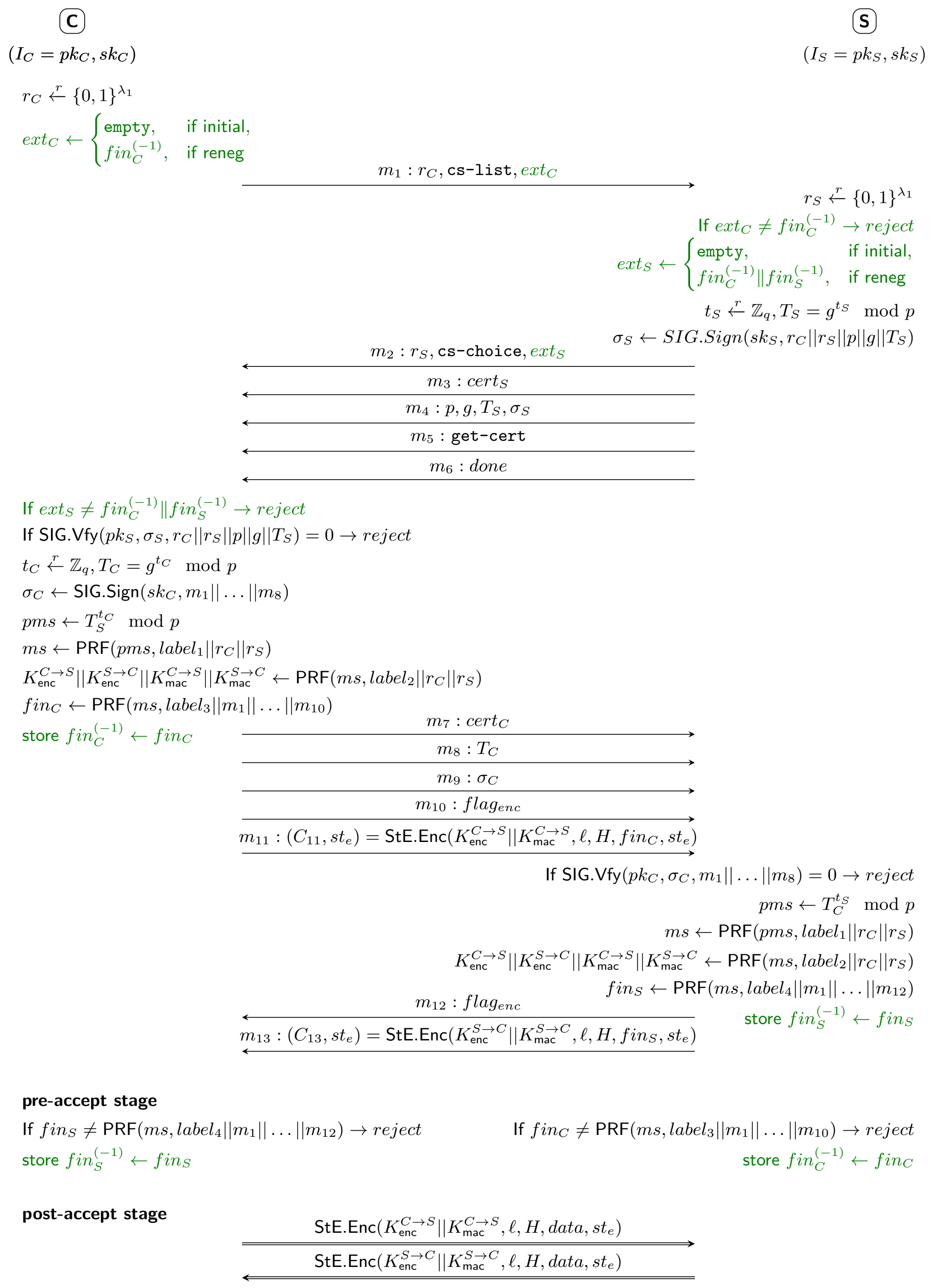
TLS allows parties to resume previous sessions [DR08, §F.1.4] without performing a full handshake. If the client wishes to resume a previous session in a new handshake, it sends the `session_id` from the previous session in its `ClientHello` message, as well as new random nonces. If the server is willing to resume the requested session, then the parties retrieve the master secret value used in previous sessions and use that master secret, along with the new random nonces, to derive new session keys.

From a high-level perspective, session resumption can be viewed as a form of renegotiation, since short-term values from the first handshake are used in subsequent handshakes. Thus, we can examine TLS session resumption in the context of our various ACCE models, viewing the "resumed session" as a new phase of the original session. We find that TLS session resumption does not satisfy our definition of a secure renegotiable ACCE (Definition 5) but can be shown to be a secure multi-phase ACCE (Definition 4).

A secure renegotiable ACCE protocol requires that, upon successful renegotiation, both parties have the same view of all previous handshake transcripts and record layer transcripts in that session. TLS session resumption does not provide that guarantee. Since in TLS session resumption the new session keys are always derived from the original master secret and new random nonces — but the master secret is never updated — parties who resume multiple sessions use the same master secret in every resumption. As a result, TLS session resumption makes no promises about the order in which multiple sessions were resumed, so an adversary can cause parties to have different views of the order of the phases.

A secure multi-phase ACCE protocol requires that whenever a party successfully renegotiates, its peer has some matching handshake transcript. It is straightforward to see TLS session resumption does provides thi, assuming TLS is itself a secure ACCE.

D `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` protocol with renegotiation extensions



$\boxed{C}$ $(I_C = pk_C, sk_C)$

$\boxed{S}$ $(I_S = pk_S, sk_S)$

C: $r_C \xleftarrow{r} \{0,1\}^{\lambda_1}$

C: $ext_C \leftarrow \begin{cases} \texttt{empty}, & \text{if initial,} \\ fin_C^{(-1)}, & \text{if reneg} \end{cases}$

C → S: $m_1 : r_C, \texttt{cs-list}, ext_C$

S: $r_S \xleftarrow{r} \{0,1\}^{\lambda_1}$

S: If $ext_C \neq fin_C^{(-1)} \rightarrow reject$

S: $ext_S \leftarrow \begin{cases} \texttt{empty}, & \text{if initial,} \\ fin_C^{(-1)} \| fin_S^{(-1)}, & \text{if reneg} \end{cases}$

S: $t_S \xleftarrow{r} \mathbb{Z}_q, T_S = g^{t_S} \mod p$

S: $\sigma_S \leftarrow SIG.Sign(sk_S, r_C||r_S||p||g||T_S)$

S → C: $m_2 : r_S, \texttt{cs-choice}, ext_S$

S → C: $m_3 : cert_S$

S → C: $m_4 : p, g, T_S, \sigma_S$

S → C: $m_5 : \texttt{get-cert}$

S → C: $m_6 : done$

C: If $ext_S \neq fin_C^{(-1)} \| fin_S^{(-1)} \rightarrow reject$

C: If $\mathsf{SIG.Vfy}(pk_S, \sigma_S, r_C||r_S||p||g||T_S) = 0 \rightarrow reject$

C: $t_C \xleftarrow{r} \mathbb{Z}_q, T_C = g^{t_C} \mod p$

C: $\sigma_C \leftarrow \mathsf{SIG.Sign}(sk_C, m_1||\ldots||m_8)$

C: $pms \leftarrow T_S^{t_C} \mod p$

C: $ms \leftarrow \mathsf{PRF}(pms, label_1||r_C||r_S)$

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

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

C: store $fin_C^{(-1)} \leftarrow fin_C$

C → S: $m_7 : cert_C$

C → S: $m_8 : T_C$

C → S: $m_9 : \sigma_C$

C → S: $m_{10} : flag_{enc}$

C → S: $m_{11} : (C_{11}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{mac}}^{C\to S}, \ell, H, fin_C, st_e)$

S: If $\mathsf{SIG.Vfy}(pk_C, \sigma_C, m_1||\ldots||m_8) = 0 \rightarrow reject$

S: $pms \leftarrow T_C^{t_S} \mod p$

S: $ms \leftarrow \mathsf{PRF}(pms, label_1||r_C||r_S)$

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

S: $fin_S \leftarrow \mathsf{PRF}(ms, label_4||m_1||\ldots||m_{12})$

S: store $fin_S^{(-1)} \leftarrow fin_S$

S → C: $m_{12} : flag_{enc}$

S → C: $m_{13} : (C_{13}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{S\to C}, \ell, H, fin_S, st_e)$

pre-accept stage

C: If $fin_S \neq \mathsf{PRF}(ms, label_4||m_1||\ldots||m_{12}) \rightarrow reject$

C: store $fin_S^{(-1)} \leftarrow fin_S$

S: If $fin_C \neq \mathsf{PRF}(ms, label_3||m_1||\ldots||m_{10}) \rightarrow reject$

S: store $fin_C^{(-1)} \leftarrow fin_C$

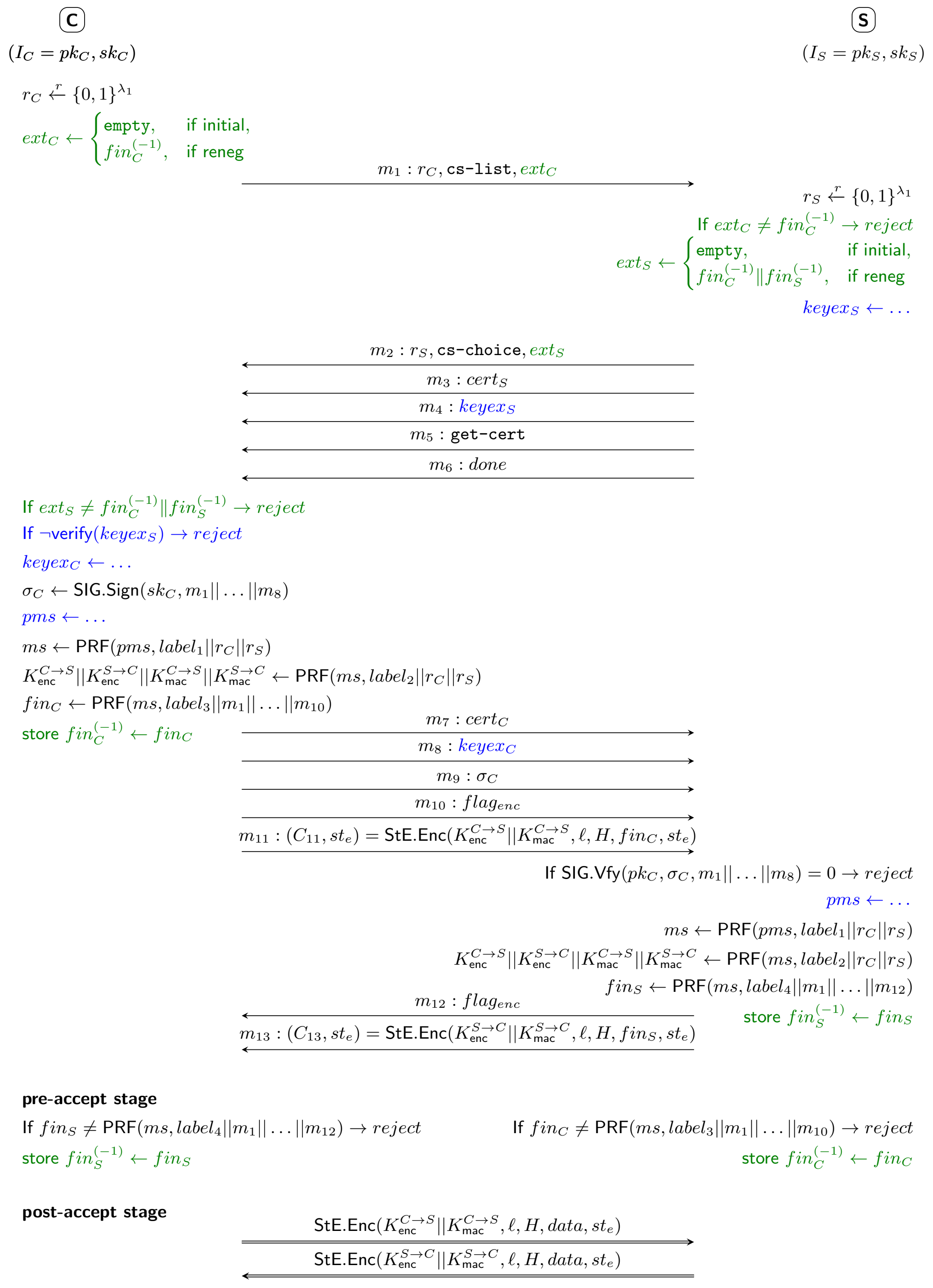
post-accept stage

C ⇒ S: $\mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S}||K_{\mathsf{mac}}^{C\to S}, \ell, H, data, st_e)$

S ⇒ C: $\mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C}||K_{\mathsf{mac}}^{S\to C}, \ell, H, data, st_e)$

Figure 4: TLS handshake for `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` ciphersuite with client authentication and SCSV / RIE renegotiation countermeasures

E Generic TLS protocol with renegotiation extensions



$\boxed{\mathsf{C}}$
$(I_C = pk_C, sk_C)$
$\boxed{\mathsf{S}}$
$(I_S = pk_S, sk_S)$
$r_C \xleftarrow{r} \{0,1\}^{\lambda_1}$
$ext_C \leftarrow \begin{cases} \texttt{empty}, & \text{if initial,} \\ fin_C^{(-1)}, & \text{if reneg} \end{cases}$
$m_1 : r_C, \texttt{cs-list}, ext_C$
$r_S \xleftarrow{r} \{0,1\}^{\lambda_1}$
If $ext_C \neq fin_C^{(-1)} \to reject$
$ext_S \leftarrow \begin{cases} \texttt{empty}, & \text{if initial,} \\ fin_C^{(-1)} \| fin_S^{(-1)}, & \text{if reneg} \end{cases}$
$keyex_S \leftarrow \ldots$
$m_2 : r_S, \texttt{cs-choice}, ext_S$
$m_3 : cert_S$
$m_4 : keyex_S$
$m_5 : \texttt{get-cert}$
$m_6 : done$
If $ext_S \neq fin_C^{(-1)} \| fin_S^{(-1)} \to reject$
If $\neg\mathsf{verify}(keyex_S) \to reject$
$keyex_C \leftarrow \ldots$
$\sigma_C \leftarrow \mathsf{SIG.Sign}(sk_C, m_1 || \ldots || m_8)$
$pms \leftarrow \ldots$
$ms \leftarrow \mathsf{PRF}(pms, label_1 || r_C || r_S)$
$K_{\mathsf{enc}}^{C\to S} || K_{\mathsf{enc}}^{S\to C} || K_{\mathsf{mac}}^{C\to S} || K_{\mathsf{mac}}^{S\to C} \leftarrow \mathsf{PRF}(ms, label_2 || r_C || r_S)$
$fin_C \leftarrow \mathsf{PRF}(ms, label_3 || m_1 || \ldots || m_{10})$
store $fin_C^{(-1)} \leftarrow fin_C$
$m_7 : cert_C$
$m_8 : keyex_C$
$m_9 : \sigma_C$
$m_{10} : flag_{enc}$
$m_{11} : (C_{11}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S} || K_{\mathsf{mac}}^{C\to S}, \ell, H, fin_C, st_e)$
If $\mathsf{SIG.Vfy}(pk_C, \sigma_C, m_1 || \ldots || m_8) = 0 \to reject$
$pms \leftarrow \ldots$
$ms \leftarrow \mathsf{PRF}(pms, label_1 || r_C || r_S)$
$K_{\mathsf{enc}}^{C\to S} || K_{\mathsf{enc}}^{S\to C} || K_{\mathsf{mac}}^{C\to S} || K_{\mathsf{mac}}^{S\to C} \leftarrow \mathsf{PRF}(ms, label_2 || r_C || r_S)$
$fin_S \leftarrow \mathsf{PRF}(ms, label_4 || m_1 || \ldots || m_{12})$
store $fin_S^{(-1)} \leftarrow fin_S$
$m_{12} : flag_{enc}$
$m_{13} : (C_{13}, st_e) = \mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C} || K_{\mathsf{mac}}^{S\to C}, \ell, H, fin_S, st_e)$
pre-accept stage
If $fin_S \neq \mathsf{PRF}(ms, label_4 || m_1 || \ldots || m_{12}) \to reject$
store $fin_S^{(-1)} \leftarrow fin_S$
If $fin_C \neq \mathsf{PRF}(ms, label_3 || m_1 || \ldots || m_{10}) \to reject$
store $fin_C^{(-1)} \leftarrow fin_C$
post-accept stage
$\mathsf{StE.Enc}(K_{\mathsf{enc}}^{C\to S} || K_{\mathsf{mac}}^{C\to S}, \ell, H, data, st_e)$
$\mathsf{StE.Enc}(K_{\mathsf{enc}}^{S\to C} || K_{\mathsf{mac}}^{S\to C}, \ell, H, data, st_e)$

Figure 5: Generic TLS handshake protocol with SCSV / RIE renegotiation countermeasures