On the Security of TLS Renegotiation

Florian Giesen
Horst Görtz Institut
Ruhr-Universität Bochum, Germany
florian.giesen@rub.de

Florian Kohlar
Horst Görtz Institut
Ruhr-Universität Bochum, Germany
florian.kohlar@rub.de

Douglas Stebila
Queensland University of Technology
Brisbane, Australia
stebila@qut.edu.au

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 renegotiation. Despite its widespread use, only recently has the full TLS protocol been proven secure, and only the core cryptographic protocol with no additional features. These additional features have been the cause of several 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. We show generically that the proposed fixes for TLS offer good protection against renegotiation attacks, and give a simple new countermeasure that provides renegotiation security for TLS even in the face of stronger adversaries.

Categories and Subject Descriptors

C.2.0 [**Computer–Communication Networks**]: General —*security and protection*

Keywords

Transport Layer Security (TLS); renegotiation; security models; key exchange

1. INTRODUCTION

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, used to agree on security parameters, establish a secret key, and authenticate the parties; and a *record layer* protocol, used to send encrypted data.

Despite the importance of TLS, progress on formally modelling the security of TLS has been slow. A technicality of TLS prevents it from being proven secure in standard authenticated key exchange (AKE) models: in AKE, the session key must be indistinguishable from a random key of the same length. However, the final handshake message of the TLS protocol is encrypted under the session key, so an adversary can distinguish the session key from a random key by trying to verify the final handshake message. Some analyses [17, 22] have shown that a truncated form of the TLS handshake is AKE-secure. Others [12] deal with a substantially weaker security requirement, namely unauthenticated key agreement. Krawczyk [19] analyzed a variant of the TLS record layer.

Only very recently have analyses of unmodified TLS functionality appeared. Paterson *et al.* [23] showed that TLS's MAC-then-encode-then-encrypt record layer when used with CBC encryption (with certain length restrictions) satisfies *length-hiding authenticated encryption (LHAE)*. Jager *et al.* [16] 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. Ciphersuites based on RSA key transport and static Diffie–Hellman have since been shown ACCE-secure by both Kohlar *et al.* [18] and Krawczyk *et al.* [20]. (An alternative modular approach to proving the full security of TLS was given by Brzuska *et al.* [5].)

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 August 2013, over 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. Record layer communication can be compressed. 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. There have been many attacks on TLS over the years, such Bleichenbacher's attack [4] and others involving padding, and Ray and Dispensa's renegotiation attack [24], all of which exploit flaws outside the core cryptographic primitives of TLS.

In this paper, we focus on renegotiation, which allows two parties to either (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 [10] and the recent thesis of Gelashvili [13], which uses the Scyther tool to automatically identify the TLS renegotiation attack. Bhargavan *et al.* [2] implement TLS supporting a variety of ciphersuites and define an application programming interface for TLS which differentiates between renegotiated phases; using typechecking, the implementation is shown secure according to a formal specification, albeit with a restricted adversary who cannot corrupt session keys.

1.1 The TLS Renegotiation Issue

All versions of TLS [7, 8, 9], and SSL v3 [11] before it, support optional 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 [24] described a man-in-the-middle attack that exploits how certain TLS-reliant applications—such as HTTP over TLS [25]—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 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.

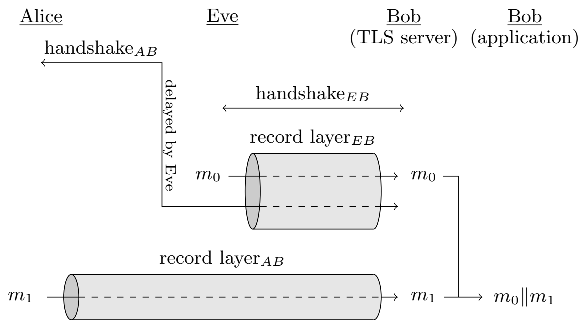


Figure 1: Ray and Dispensa's man-in-the-middle renegotiation attack on TLS-reliant applications

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 [24] and SMTPS [28], concatenating m_0 and m_1 and treating them as coming from the same party in the same context. For example, if Eve sends the HTTP request m_0 and Alice sends the HTTP request m_1, where

m_0 = "`GET /orderPizza?deliverTo=123-Fake-St` ↩
`X-Ignore-This: `"
m_1 = "`GET /orderPizza?deliverTo=456-Real-St` ↩
`Cookie: Account=111A2B`"

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

$m_0 \| m_1$ = "`GET /orderPizza?deliverTo=123-Fake-St` ↩
`X-Ignore-This: GET /orderPizza`
`?deliverTo=456-Real-St` ↩
`Cookie: Account=111A2B`"

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, so this line is ignored. However, the following line, Alice's account cookie, is still processed. Eve is able to have the pizza delivered to herself but paid for by Alice.

It should be noted that Ray and Dispensa's attack works for both server-only authentication and mutual authentication modes of TLS: the use of client certificates in general does not prevent the attack [24, 28].

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 [26] to provide countermeasures to this attack, with the goal of applicability to SSLv3.0 and TLS versions 1.0–1.2. Two countermeasures were standardized: the *Signalling Ciphersuite Value (SCSV)* and the *Renegotiation Information Extension (RIE)*. In RIE, the parties include the key confirmation value from the previous handshake in a `ClientHello`/`ServerHello` extension [3], demonstrating they have the same view of the previous handshake, or a distinguished null value if not renegotiation. SCSV is a slight modification that is more compatible with buggy implementations. A diagram showing the message flow for a generic TLS ciphersuite with SCSV/RIE countermeasures appears in Figure 2 in Appendix B. According to one survey [27], as of July 2013, 82% of TLS-enabled websites support SCSV/RIE, with 9% still supporting insecure renegotiation and 9% not supporting renegotiation.

1.3 Contributions

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 start from the ACCE model, rather than AKE security models. 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.[1] This is qualitatively different than simply having multiple sessions, since short-term values from one phase of a session may be used in the renegotiation for the next phase, whereas multiple sessions only share long-term values. Each oracle maintains state and encryption/MAC keys for each phase. Like in TLS, our formalism allows control messages to be sent on the encrypted channel. Our extension to the ACCE model also models server-only authentication in addition to mutual authentication.

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, generically, TLS with the SCSV/RIE countermeasures of RFC 5746 [26] is a *weakly secure renegotiable ACCE protocol.* In this slightly weaker—but still quite reasonable—model, the adversary is slightly restricted in the previous secrets she is allowed to reveal.

Our approach for proving the renegotiable security of TLS with SCSV/RIE countermeasures is modular. We *cannot* generically prove that an ACCE-secure TLS ciphersuite is, with SCSV/RIE, a weakly secure renegotiable ACCE, because the protocol itself is modified by including SCSV/RIE and hence a black-box approach does not work. Instead, we consider *tagged TLS* where an arbitrary tag can be provided as an extension. Via a chain of results and models, we show that if a tagged TLS ciphersuite is secure in an ACCE variant where `Finished` messages are revealed, then that TLS ciphersuite with SCSV/RIE is a weakly secure renegotiable ACCE protocol. This provides a generic justification for the security of SCSV/RIE. Proving that TLS ciphersuite secure in this tagged variant model seems to be almost no harder than a proof that it is ACCE-secure; we only needed to change a few lines from the ACCE security proof of `TLS_DHE_DSS_` [16].

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

[1]Note that TLS standards use different words. We say a single *session* can have multiple *phases*; the TLS standards refer to a single *connection* having multiple *sessions*.

	Secure multi-phase ACCE (Defn. 3)	**Weakly secure renegotiable ACCE** (Defn. 5)	**Secure renegotiable ACCE** (Defn. 4)
Secure against Ray–Dispensa-type attacks	×	✓	✓[1]
`TLS` w/o countermeasures	—	× (§3)	× (§3)
with SCSV/RIE (if tagged-ACCE-fin-secure)	—	✓ (Thm. 1)	× (§4.1)
with new countermeasure (if tagged-ACCE-fin-secure)	✓ (Thm. 4)	✓ (Thm. 4)	✓ (Thm. 4)

[1] For secure renegotiable ACCE, security holds even if session keys from one phase are revealed while that phase is active.

Table 1: Summary of security notions and results on TLS

message injections by the adversary. Though the ability to learn session 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. This countermeasure can be seen as providing *record layer recognition*, confirming that both parties have the same view of all communicated messages, rather than just *handshake recognition* as in the SCSV/RIE countermeasure.

Table 1 summarizes our results on TLS and renegotiation countermeasures. While our theorems are specific to TLS, our definitional framework is suitable for any channel establishment protocol that involves renegotiation, and could for example be used to check if renegotiation in the Secure Shell (SSH) protocol really is secure [21].

On composability and the choice of ACCE. It would be desirable to prove the security of the TLS renegotiation countermeasures via some kind of composability framework, such as universal composability or the game-based composability framework of Brzuska *et al.* [5]. Unfortunately, this is not possible with existing security definitions for TLS. The TLS renegotiation countermeasures are not achieved by *composing* in a black-box manner one protocol or primitive with another. Instead, the SCSV/RIE countermeasure looks inside the protocol and changes it in a white-box way: it *modifies* the messages sent by the protocol, and *re-uses* an internal value. Thus we cannot make use of existing security results in a black-box compositional way. Our approach is the "next best thing": we modify an existing security definition (ACCE) in what seems to be a minimal way, adding just enough "hooks" to get at the internal values needed to modify and re-use the required values for the SCSV/RIE countermeasure. We are then able to prove in a fully generic way that any TLS protocol that satisfies this slightly modified ACCE notion with hooks is, when using the SCSV/RIE countermeasure, secure against renegotiation attacks. Since the hooks added are quite small, it is not much work to change a proof that a TLS ciphersuite is ACCE secure to show that it satisfies this slightly modified ACCE notion as well.

Of the two existing definitional approaches for proving the full security of the TLS protocol [16, 5], we chose the ACCE approach over the game-based composability approach because renegotiation in TLS makes extensive use of the interplay between the handshake and record layer.

Moreover, the modifications we make to the execution environment to enable analysis of renegotiable ACCE protocols can be adapted to capture renegotiation and rekeying in other types of protocols such as authenticated key exchange.

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. Essentially, a multi-phase protocol can have many key exchanges—each called a *phase*—linked to a single session. Our definition builds on the ACCE definition of Jager *et al.* [16], which combined the Bellare–Rogaway model for authenticated key exchange [1] with a Jager *et al.*'s stateful variant of Paterson *et al.*'s length-hiding authenticated encryption [23].

Notation. If S is a set, $x \xleftarrow{\$} S$ 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 1-indexed array $\texttt{phases}$ and $|\texttt{phases}|$ denotes the number of entries in the array. $\pi_A^s.x$ denotes variable x stored in oracle instance π_A^s.

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 *(multi-phase) authentication* if a party accepts in a phase with uncorrupted long-term keys, but no matching phase exists at the peer.
- An adversary breaks *confidentiality/integrity* if it can guess the bit b involved in a confidentiality/integrity experiment similar to stateful length-hiding authenticated encryption.

Our main security definition is a *secure renegotiable ACCE protocol*, which strengthens the authentication notion: parties should successfully renegotiate only when they have exact same view of everything that happened before.

- 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.*

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 the SCSV/RIE countermeasure for TLS generically provides 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.

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

2.2 Execution Environment

Parties. The environment consists of n_{pa} parties, $\{P_1, \ldots, P_{n_{\text{pa}}}\}$. Each party P_A is a potential protocol participant, and has a list of n_{ke} long-term key pairs $(pk_{A,1}, sk_{A,1}), \ldots, (pk_{A,n_{\text{ke}}}, sk_{A,n_{\text{ke}}})$. 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. It is common in AKE security models to assume ideal distribution of long-term public keys for simplicity [1, 6, 16].

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 $s \in [1, n_{\text{se}}]$ 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.$\pi_A^s.d = \pi_B^t$. As well, the variable $\rho \in \{\mathsf{Client}, \mathsf{Server}\}$ records the role of the oracle. Parties can play the role of the client in some sessions and of the server in other sessions, but their role is fixed across all phases within a session.

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 or server-only authentication; and a *post-accept*, or "record layer", stage, which provides a stateful communication channel with confidentiality and integrity. A list $\pi_A^s.\texttt{phases}$ of different phase states is maintained; we sometimes use the notation $\pi_A^{s,\ell}$ for $\pi_A^s.\texttt{phases}[\ell]$. There can be at most n_{ph} phases per oracle. The last entry of $\pi_A^s.\texttt{phases}$ contains the state of the current phase, which may still be in progress. Each entry $\pi_A^s.\texttt{phases}[\ell]$ 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 as being used for its peer in that phase[2],
- $\omega \in \{0, 1\}$, denoting the authentication mode used, where 0 indicates that server-only authentication is used in that phase and 1 indicates mutual authentication,
- Δ, a counter used to keep track of the current status of the protocol execution,
- α, either $\texttt{accept}$, $\texttt{reject}$, 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 (plaintext) messages sent and received by π_A^s during the pre-accept stage of that phase,

[2]One of the public keys may remain empty, if no client authentication is requested.

- RT_s and RT_r, the transcripts of all ciphertexts sent and received (respectively) 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).

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. The variable ω is set at some point during (or before) the protocol execution, depending on the protocol specification (e.g.in case of TLS, the server can send the message `CertificateRequest` to request client, i.e. mutual, authentication, otherwise server-only authentication is used). 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 variables RT_s and RT_r 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), the plaintext messages are appended to variable T in the new entry of the phase log, and ciphertexts are appended to 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. *[16].*

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 (Corrupt), learn session keys (Reveal), and control sending and receiving of arbitrary messages on the encrypted record layer (Encrypt/Decrypt) using a stateful symmetric encryption scheme StE [23].

- $\mathsf{Send}(\pi_A^s, m)$: 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:
  - $m = (\texttt{newphase}, pk, \omega)$ triggers an oracle to initiate renegotiation of a new phase (or new session if first phase). Note that the action here may vary based on the role of the party: for example, when renegotiating in TLS, a client would prepare a new `ClientHello` message, encrypt it by calling the Encrypt oracle below, and then return the ciphertext to the adversary for delivery; a server would correspondingly prepare an encrypted `ServerHelloRequest` message.
  - $m = (\texttt{ready}, pk, \omega)$ activates a (server) oracle to use the public key pk in its next phase.

  For the above control messages, pk indicates the long-term public key pk the oracle should use in the phase and ω indicates the authentication mode to use; 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 [6]. Delivery of encrypted messages in the post-accept stage are handled by the 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 obtain the party's internal state nor register rogue keys.
- $\mathsf{Reveal}(\pi_A^s, \ell)$: Oracle π_A^s responds with the key(s) $\pi_A^s.\texttt{phases}[\ell].k$ used in phase ℓ, or $\emptyset$ if no such value exists. Since the TLS record layer is unidirectional, there are both encryption and decryption keys, and for most ciphersuites also MAC keys, so all 4 keys ($K_{\mathsf{enc}}^{C\to S}$, $K_{\mathsf{enc}}^{S\to C}$, $K_{\mathsf{mac}}^{C\to S}$, $K_{\mathsf{mac}}^{S\to C}$) would be revealed, though one could refine if desired.
- $\mathsf{Encrypt}(\pi_A^s, ctype, m_0, m_1, len, hd)$: This query takes as input a content type $ctype$, messages m_0 and m_1, a length len, and header data hd. Content type `control` is used for handshake messages. The adversary cannot query this oracle with $ctype =$ `control`. Through an abuse of notation, we allow the party itself to call this oracle with `control` to encrypt protocol messages that must be sent encrypted; this abuse of notation allows the party to construct encrypted protocol messages while all aspects of the security experiment remain synchronized. 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. Encrypt depends on the random bit b sampled by π_A^s at the beginning of the current phase. It maintains a counter u initialized to 0 and an encryption state st_e, and proceeds as follows:
  1. $u_A^s \leftarrow u_A^s + 1$
  2. If ($ctype =$ `control`) AND caller is not π_A^s, then return $\perp$
  3. $(C^{(0)}, st_e^{(0)}) \overset{\$}{\leftarrow} \mathsf{StE.Enc}(k, len, hd, ctype\|m_0, st_e)$
  4. $(C^{(1)}, st_e^{(1)}) \overset{\$}{\leftarrow} \mathsf{StE.Enc}(k, len, hd, ctype\|m_1, st_e)$
  5. If $(C^{(0)} = \perp)$ OR $(C^{(1)} = \perp)$, then return $\perp$
  6. $(C_A^s[u_A^s], st_e) \leftarrow (C^{(b_s^A)}, st_e^{(b_s^A)})$
  7. Return $C_A^s[u_A^s]$
- $\mathsf{Decrypt}(\pi_A^s, C, hd)$: This query takes as input a ciphertext C and header data hd. If π_A^s has not accepted in the current phase, then it returns $\perp$. Decrypt maintains a counter v and a switch $\mathsf{diverge}$, both initialized to 0, and a decryption state st_d, and proceeds as described below. 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 by calling $\mathsf{Encrypt}$ according to the protocol specification. Technically it proceeds as follows:

1. $(B, t) \leftarrow \pi_A^s.d$, $v_A^s \leftarrow v_A^s + 1$, $m' \leftarrow \emptyset$
2. $(ctype \| m, st_d) = \mathsf{StE.Dec}(k, hd, C, st_d)$
3. If $(v_A^s > u_B^t)$ OR $(C \neq C_B^t[v_A^s])$, then $\mathsf{diverge} \leftarrow 1$
4. If $(b_A^s = 1)$ 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')

Remark 2. *Note that k may be a 'multi-part' key with different parts for encryption, decryption, and MAC; we assume* $\mathsf{StE.Enc}$ *and* $\mathsf{StE.Dec}$ *know which parts to use. Also note that the 'protocol response for m' may be encrypted by the party internally making an* $\mathsf{Encrypt}$ *call.*

The behaviour of the $\mathsf{Decrypt}$ oracle in this combined definition for confidentiality and integrity can be somewhat difficult to understand. It extends that of stateful length-hiding authenticated encryption as originally defined by Paterson *et al.* [23].

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 [16]) 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 defined 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, $\mathsf{Corrupt}$, Reveal, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ queries to the oracles and eventually terminates.

Definition 1 (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 $\ell, \ell' \in [1, n_{\text{ph}}]$ for which $\pi_A^s.\mathsf{phases}[\ell].T$ and $\pi_B^t.\mathsf{phases}[\ell'].T$ are matching conversations, it holds that $\pi_A^s.\mathsf{phases}[\ell].\alpha = \pi_B^t.\mathsf{phases}[\ell'].\alpha = \mathtt{accept}$, $\pi_A^s.\mathsf{phases}[\ell].\omega = \pi_B^t.\mathsf{phases}[\ell].\omega$ and $\pi_A^s.\mathsf{phases}[\ell].k = \pi_B^t.\mathsf{phases}[\ell'].k$.*

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 $\mathsf{Encrypt}/\mathsf{Decrypt}$ 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.* [16], this notion ensures forward security: corrupting long-term secret keys after completion of a session should not impact confidentiality/integrity of messages.

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

*then we say $\mathcal{A}$ (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 as a stepping stone in our generic result in Section 4. 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 long-term secret keys before the phase accepted).

Definition 3 (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. $\pi_A^s.\mathsf{phases}[\ell].\alpha = \mathtt{accept}$*; and*
A2. *$\mathcal{A}$ did not query* $\mathsf{Corrupt}(P_A, \pi_A^s.\mathsf{phases}[\ell].pk)$ *before π_A^s accepted in phase ℓ; and*
A3. *$\mathcal{A}$ did not query* $\mathsf{Corrupt}(P_B, \pi_A^s.\mathsf{phases}[\ell].pk')$ *before π_A^s accepted in phase ℓ; and*
A4. *if $\pi_A^s.\mathsf{phases}[\ell].\omega = 0$ then $\pi_A^s.\rho = \mathsf{Client}$; and*
A5. *$\mathcal{A}$ did not query* $\mathsf{Reveal}(\pi_B^t, \ell')$ *before π_A^s accepted in phase ℓ for any ℓ' such that $\pi_B^t.\mathsf{phases}[\ell'].T$ is a matching conversation to $\pi_A^s.\mathsf{phases}[\ell].T$; and*
M. *there is no ℓ' such that $\pi_B^t.\mathsf{phases}[\ell'].T$ is a matching conversation to $\pi_A^s.\mathsf{phases}[\ell].T$*

*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. 2) or authentication (as above) of Π.*

In **A1** and **M** we redefine the $\texttt{NoMatch}$-condition from [1]. In **A2** we exclude leaking of the secret long-term keys of the accepting party (necessary for example to counter key compromise impersonation attacks). In **A3** we exclude corruptions of the peer. In **A4** (only for server-only authentication), we ensure that the adversary only wins by making a client-oracle maliciously accept. In **A5** we exclude trivial attacks that exist for protocols with explicit key confirmation and probabilistic computations under the negotiated key.

The secure multi-phase ACCE definition when limited to a phase per session and a single key pair per party ($n_{\mathrm{ph}} = n_{\mathrm{ke}} = 1$) collapses to an extension of the original ACCE definition, the extension being support for server-only authentication.

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 with our proposed countermeasure satisfies this definition.

Definition 4 (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–A5 *as in Definition 3 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 $\ell < \ell^*$, $\pi_A^s.\mathtt{phases}[\ell].T \| RT_s \| RT_r \neq \pi_B^t.\mathtt{phases}[\ell].T \| RT_r \| RT_s$;*

*we say $\mathcal{A}$ (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. 2) or renegotiation authentication (as defined above) of* Π.

2.3.4 Weakly secure renegotiable ACCE.

Unfortunately, TLS with SCSV/RIE does not meet Def. 4 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 SCSV/RIE 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. Our desire to characterize the renegotiable security of the SCSV/RIE countermeasure 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, no adversary should 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 5 (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. 4, as well as the following additional conditions are satisfied:*

A6. *$\mathcal{A}$ 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*

A7. *$\mathcal{A}$ 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. 2) or weak renegotiation authentication (as defined above) of* Π.

Remark 3. *While conditions **A6** and **A7** prohibit the adversary from revealing encryption keys of previous phases while active* for the purposes of breaking authentication*, the confidentiality/integrity aspect of Def. 5 still places no such restriction on previous encryption keys being revealed.*

Remark 4. *The chain of implications between Defns. 3–5 is as follows: secure renegotiable ACCE protocol (Defn. 4) $\Longrightarrow$ weakly secure renegotiable ACCE (Defn. 5) $\Longrightarrow$ secure multi-phase ACCE (Defn. 3) $\Longrightarrow$ ACCE protocol [16]. The separations are strict, as seen in Table 1.*

3. TLS RENEGOTIATION (IN)SECURITY

We now 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.

Jager *et al.* [16] in their full version described how to map TLS into the ACCE model. We highlight a few components of that mapping, and the alterations needed for renegotiation.

Oracles generally respond to Send, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ queries as specified by the TLS handshake and record layer protocols. The Send control message $m = (\mathtt{newphase}, pk)$ 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 $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ queries, we use a content type field *ctype* that corresponds to the `ContentType` field of the `TLSPlaintext` data type in the TLS record layer specification [9, §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

Recall the TLS renegotiation attack by Ray and Dispensa [24], as described previously in Figure 1 on page . 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 5 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 handshakes/record layer transcripts do not match Alice's). Thus TLS without countermeasures is not a weakly secure or secure renegotiable ACCE.

4. RENEGOTIATION SECURITY OF TLS WITH SCSV/RIE COUNTERMEASURE

In this section we analyze the security of TLS with the SCSV/RIE countermeasures proposed in RFC 5746 [26]. We first see, in Section 4.1, that the SCSV/RIE countermeasures are not enough to prove that TLS satisfies our strongest notion, a secure renegotiable ACCE (Defn. 4).

Our goal will be to show that TLS with the SCSV/RIE countermeasures is a weakly secure renegotiable ACCE. Ideally, we would do so generically, with a result saying something like "If a TLS ciphersuite is a secure ACCE, then that TLS ciphersuite with SCSV/RIE is a weakly secure renegotiable ACCE." As noted in the introduction, we do so generically since the protocol is *modified* to include the countermeasure values in the `ClientHello` and `ServerHello` messages, and thus we cannot make use of the ACCE security of the particular TLS ciphersuite in a black-box way. Moreover, we must ensure that revealing the `Finished` values from the previous handshake does not impact its security. Although these barriers prevent a generic black-box result, a white-box examination of the proof details of `TLS_DHE_DSS_` [16] finds that only small changes are needed in the proof.

We will provide a sequence of definitions and results that justifies the security of the SCSV/RIE countermeasure.

1. Define a modified ACCE security model, called *tagged-ACCE-fin*, specific to TLS, in which the adversary can reveal `Finished` messages after the handshake completes and supply tags to be used in extensions.
2. Define *tagged TLS* as a modification of a standard ciphersuite in which arbitrary opaque data can be placed in an extension field in the `ClientHello` and `ServerHello` messages.
3. Explain how the existing proof of that `TLS_DHE_DSS_` is ACCE secure can be modified in a very minor way to show that tagged `TLS_DHE_DSS_` is tagged-ACCE-fin-secure.
4. Show that, if a TLS ciphersuite is tagged-ACCE-fin secure, then that TLS ciphersuite with SCSV/RIE is a secure multi-phase ACCE.
5. Show that, if a TLS ciphersuite with SCSV/RIE is a secure multi-phase ACCE, then it is also a weakly secure renegotiable ACCE.

Combined, these results yield (a) a general result justifying the security of the SCSV/RIE countermeasure, and (b) that `TLS_DHE_DSS_` with SCSV/RIE countermeasures is a weakly secure renegotiable ACCE.

4.1 TLS with SCSV/RIE is not a secure renegotiable ACCE

Definition 4 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 4 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. SCSV/RIE meets its original goal of authenticating the previous handshake.

4.2 Tagged-ACCE model and tagged TLS

In this section we introduce a variant of the ACCE model from which we can prove a generic result on the renegotiable security of TLS with countermeasures. In our models, the tag can be an arbitrary string. For our purpose of analyzing TLS with SCSV/RIE countermeasures, the tag will be the previous phase's `Finished` messages.

4.2.1 Tagged-ACCE security model

The *tagged-ACCE* security model is an extension of the ACCE security model to allow arbitrary tags as follows. Since the original ACCE definition of Jager *et al.* [16] does not support server-only authentication, while our definition allows both authentication modes, we extend the ACCE definition implied by limiting multi-phase ACCE (Definition 3) to a single phase ($n_{\mathrm{ph}} = 1$) and at most one public key per party ($n_{\mathrm{ke}} = 1$).

The phases log `phases` is extended with an additional per-phase variable *tag*.

- $\mathsf{Send}(\pi^s_A, m)$. The adversary can specify an arbitrary tag during session initialization.
  - If $m = (\mathtt{newphase}, \omega, tag)$, the party sets its internal variable $\rho \leftarrow \mathsf{Client}$, sets authentication mode ω, stores tag, and responds with the first protocol message.
  - If $m = (\mathtt{ready}, \omega, tag)$, the party sets $\rho \leftarrow \mathsf{Server}$, authentication mode ω, stores tag, and responds with the next protocol message, if any.

The freshness and winning conditions of tagged-ACCE are unchanged from ACCE.

4.2.2 Tagged-ACCE-fin security model

We will work with a further variant, *tagged-ACCE-fin*, which is not a fully general security model but instead is tied specifically to generic TLS protocols of the form given in Figure 2. It adds the following query:

- $\mathsf{RevealFin}(\pi_A^s)$: If $\alpha = \mathtt{accept}$, then return the fin_C and fin_S values sent/received by the queried oracle. Return $\emptyset$ otherwise.

The following queries are modified:

- $\mathsf{Encrypt}(\pi_A^s, ctype, m_0, m_1, len, hd)$: The adversary is not prevented from querying with $ctype = \mathtt{control}$.
- $\mathsf{Decrypt}(\pi_A^s, C, hd)$: No semantic meaning is associated with $ctype = \mathtt{control}$ messages. In other words, line 5 of $\mathsf{Decrypt}$ is removed.

We extend the Encrypt- and Decrypt-queries to allow the adversary to send and receive messages on the encrypted channel with content type `control`. The freshness and winning conditions of tagged-ACCE-fin are as in ACCE.

Remark 5. *Revealing the `Finished` messages is very specific to the TLS protocol family and is not necessarily relevant for other protocols. Imagine, for example, a variant of the SCSV/RIE countermeasure where a separate hash of the complete transcript as it was sent over the channel is used as an authenticator. Since this value can be computed by any passive adversary, leaking this value could not affect security.*

4.2.3 Tagged TLS

Figure 2 shows a generic TLS ciphersuite, along with the SCSV/RIE extensions denoted in green with a dagger. By *tagged TLS*, we mean the generic TLS ciphersuite from Figure 2, without any of the SCSV/RIE extensions shown in green, but where an arbitrary string can be placed in the ext_C and ext_S fields. In other words, it is a normal TLS ciphersuite, but with an arbitrary extension field that just carries strings that are not being interpreted as having any particular meaning.

As noted in the beginning of this section, we cannot generically prove that, if a TLS ciphersuite is ACCE-secure, then the tagged version of that ciphersuite is tagged-ACCE- or tagged-ACCE-fin-secure, as we have made white-box modifications to the TLS protocol in introducing the SCSV/RIE countermeasure. Thus we cannot use its security results in a black-box manner. However, in most cases, a white-box approach, where the actual security proof is modified or extended, should be possible, and even very easy. This was indeed the case when we examined tagged `TLS_DHE_DSS_`.

Theorem 1 (Informal). *Under the same assumptions on the cryptographic building blocks as in Jager* et al. *[16],* `TLS_DHE_DSS_` *is a secure tagged-ACCE-fin protocol.*

The formal theorem statement and proof are omitted due to the space limitation and appear in the full version [14]. The proof follows almost exactly the proof by Jager *et al.* [16] that `TLS_DHE_DSS_` is a secure ACCE protocol. Leaking the `Finished` messages does not affect security due to a game hop where, due pseudorandomness of the PRF, the `Finished` messages are replaced with uniformly random values independent of any information exchanged during the handshake. Including arbitrary extra data in the handshake messages does not impact security.

4.3 TLS with SCSV/RIE is multi-phase-secure

We begin by showing that including the SCSV/RIE countermeasure does not *weaken* security: putting the `Finished` messages in the `ClientHello` and `ServerHello` does not introduce any vulnerabilities. Having done so, in the next subsection we will show how including the SCSV/RIE countermeasure yields a weakly secure renegotiable ACCE.

Theorem 2. *Let Π be a generic tagged TLS ciphersuite as described in Section 4.2. Assume that Π is $(\tau, \epsilon_{\mathsf{tagged}})$-tagged-ACCE-fin-secure. Let Π' denote Π with SCSV/RIE countermeasures as described in Figure 2. For any adversary that $(\tau', \epsilon_{\mathsf{mp}})$-breaks the multi-phase ACCE security of Π' with $\tau \approx \tau'$, it holds that $\epsilon_{\mathsf{mp}} \leq 2\epsilon'$, where ϵ' is obtained from ϵ by replacing all instances of n_{pa} in ϵ with $n_{\mathrm{pa}} \cdot n_{\mathrm{ke}}$ and replacing all instances of n_{se} in ϵ with $n_{\mathrm{se}} \cdot n_{\mathrm{ph}}$. (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.)*

Due to the page limitation we only give the proof strategy here and refer to the anonymous full version [14] for the details.

Proof idea: We will construct a multi-phase ACCE simulator $\mathcal{S}$ for Π' that makes use of a tagged-ACCE-fin challenger $\mathcal{C}$ for Π. $\mathcal{S}$ will simulate every (party, public-key) pair and every (session, phase) pair with distinct parties and sessions in $\mathcal{C}$. For the most part, $\mathcal{S}$ will relay queries down to $\mathcal{C}$ and return the result. However, for queries that relate to renegotiation (Send, Decrypt), $\mathcal{S}$ carefully manages the handshake messages to transition one session in $\mathcal{C}$ to another. □

Remark 6. *A simulation similar to the one in the proof allows us to prove that TLS with SCSV/RIE countermeasures is a multi-phase ACCE protocol, even when different ciphersuites are used in different phases. The simulator interacts with a different tagged-ACCE-fin challenger for each ciphersuite; when a renegotiation inside one ciphersuite will result in a new ciphersuite, the simulator uses the* Encrypt/Decrypt *queries in the old ciphersuite to encrypt the* Send *messages from the handshake of the new ciphersuite. Unfortunately, for this multi-ciphersuite simulation to work, it is essential that public keys not be shared across ciphersuites: this technique could show that switching between an RSA-based ciphersuite and an ECDSA-based ciphersuite is safe. However, to analyze using the same RSA public key in two different ciphersuites, one would have to take an alternative approach, as it may not be possible to generically prove that re-using the same public key in two ACCE protocols is safe.*

4.4 TLS with SCSV/RIE is a weakly secure renegotiable ACCE

We are now in a position to show that the use of the SCSV/RIE countermeasure in TLS results in a weakly secure renegotiable ACCE. We will do so generically, starting from the consequence of the previous theorem: that TLS with SCSV/RIE is a secure multi-phase ACCE.

Theorem 3. *Let Π be a TLS ciphersuite with SCSV/RIE countermeasures, as described in Figure 2. If Π is a $(\tau, \epsilon_{\mathsf{mp}})$-secure multi-phase ACCE protocol, and* PRF *is a $(\tau, \epsilon_{\mathsf{prf}})$-secure pseudorandom function, then Π is a (τ, ϵ)-weakly secure renegotiable ACCE, with $\epsilon = \epsilon_{\mathsf{mp}} + \epsilon_{\mathsf{prf}}$.*

The full proof of the theorem appears in Appendix A.

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 property of 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.

We can combine Theorems 2 and 3 to obtain the central results of the paper, justifying the security of the SCSV/RIE countermeasure:

Corollary 1. *If a tagged TLS ciphersuite* Π *as described in Section 4.2 is a secure tagged-ACCE-fin protocol and* PRF *is a secure pseudorandom function, then that TLS ciphersuite* Π *with SCSV/RIE countermeasures as described in Figure 2 is a weakly secure renegotiable ACCE.* □

Corollary 2. *Under the same assumptions 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 4. 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:

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

where $ms^{(-1)}$ is the previous phase's master secret, H is a collision-resistant hash function, and $RT_s^{(-1)} \| RT_r^{(-1)}$ is the client's view of the previous phase's record layer transcript; the server uses $RT_r^{(-1)} \| RT_s^{(-1)}$ instead. 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.

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. However, the cost of the MAC computation can dominate the cryptographic cost of record layer computations [15]. The new countermeasure is only suitable for TLS communications over a reliable channel and could not be used with DTLS communications over an unreliable channel.

Alternatively, if the sLHAE scheme for the record layer is implemented as encrypt-then-MAC or MAC-then-encrypt, it should be possible to use MAC contained in the last encrypted message of the sLHAE scheme instead of the hash value computed above; this would result in no additional performance impact and would be easier to implement.

Theorem 4. *Let* Π *be a TLS ciphersuite with the original RIE countermeasures as in Figure 2 but using* ext_C *as in equation (1). If* Π *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* Π *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 3. The main difference is that, in one case, the removal of restrictions **A6** and **A7** means we can no longer reduce down to a violation of confidentiality/integrity in the multi-phase security of Π, 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.

We refer to the full version [14] for the full proof.

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 practical issue of renegotiation, in which parties update one or more aspects of their connection — 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. We have provide a generic proof that the SCSV/RIE countermeasure offers good protection against renegotiation attacks, and give a new countermeasure that provides renegotiation security even in the face of slightly stronger adversaries. In practice, the SCSV/RIE countermeasure may be good enough.

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 and the formal analysis of other complex functionality such as alerts and error messages. TLS session resumption [9, §F.1.4] is another important functionality of TLS, and it appears that our multi-phase ACCE model may be the right model in which to analyze its security, another interesting open problem. 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 discussions with Colin Boyd, Cas Cremers, Kenny Paterson, Jörg Schwenk, and the authors of the SCSV/RIE countermeasure [26], and the advice of anonymous reviewers. The research was supported by an Australian Technology Network–German Academic Exchange Service (ATN-DAAD) joint research cooperation scheme grant. D.S. was supported by an Australian Research Council (ARC) Discovery Project.

7. REFERENCES

[1] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. CRYPTO'93*, vol. 773 of *LNCS*, pp. 232–249.

[2] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P.-Y. Strub. Implementing tls with verified cryptographic security. In *IEEE Symposium on Security & Privacy*, 2013. http://mitls.rocq.inria.fr/.

[3] S. Blake-Wilson, M. Nystroem, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) extensions, June 2003. RFC 3546.

[4] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Proc. CRYPTO'98*, vol. 1462 of *LNCS*, pp. 1–12.

[5] C. Brzuska, M. Fischlin, N. P. Smart, B. Warinschi, and S. C. Williams. Less is more: Relaxed yet composable security notions for key exchange. *Int. J. Information Security*, 12(4):267–297.

[6] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Proc. CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 143–161.

[7] T. Dierks and C. Allen. The TLS protocol version 1.0, January 1999. RFC 2246.

[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.1, April 2006. RFC 4346.

[9] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) protocol version 1.2, August 2008. RFC 5246.

[10] S. Farrell. Why didn't we spot that? *IEEE Internet Computing*, 14(1):84–87, Jan.–Feb. 2010.

[11] A. O. Freier, P. Karlton, and P. C. Kocher. The Secure Sockets Layer (SSL) protocol version 3.0, August 2011. RFC 6101; republication of original SSL 3.0 specification by Netscape of November 18, 1996.

[12] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk. Universally composable security analysis of TLS. In *Proc. ProvSec 2008*, vol. 5324 of *LNCS*, pp. 313–327.

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

[14] F. Giesen, F. Kohlar, and D. Stebila. On the security of TLS renegotiation (full version), 2013. http://eprint.iacr.org/2012/630.

[15] V. Gupta, D. Stebila, S. Fung, S. C. Shantz, N. Gura, and H. Eberle. Speeding up secure web transactions using elliptic curve cryptography. In *Proc. NDSS 2004*. The Internet Society, Feb. 2004.

[16] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DHE in the standard model. In *Proc. CRYPTO 2012*, vol. 7417 of *LNCS*, pp. 273–293. Full version: http://eprint.iacr.org/2011/219.

[17] J. Jonsson and B. S. Kaliski Jr. On the security of RSA encryption in TLS. In *Proc. CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 127–142.

[18] F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DH and TLS-RSA in the standard model, 2013. http://eprint.iacr.org/2013/367.

[19] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Proc. CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 310–331.

[20] H. Krawczyk, K. G. Paterson, and H. Wee. On the security of the TLS protocol: A systematic analysis. In *Proc. CRYPTO 2013*, vol. 8042 of *LNCS*, pp. 429–448.

[21] D. Miller. SSL vulnerability and SSH, November 2009. http://lists.mindrot.org/pipermail/openssh-unix-dev/2009-November/028003.html.

[22] P. Morrissey, N. P. Smart, and B. Warinschi. A modular security analysis of the TLS handshake protocol. In *Proc. ASIACRYPT 2008*, vol. 5350 of *LNCS*, pp. 55–73.

[23] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *Proc. ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 372–389.

[24] M. Ray and S. Dispensa. Renegotiating TLS, November 2009.

[25] E. Rescorla. HTTP over TLS, May 2000. RFC 2818.

[26] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) renegotiation indication extension, February 2010. RFC 5746.

[27] Trustworthy Internet Movement. SSL Pulse, July 2013. https://www.trustworthyinternet.org/ssl-pulse/.

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

APPENDIX

A. PROOF OF THEOREM 3

PROOF. Suppose $\mathcal{A}$ breaks the weak renegotiable ACCE security of the protocol Π. We will show that either $\mathcal{A}$ breaks the multi-phase ACCE security of Π 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 Π.

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, the latter 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 Π breaks confidentiality/integrity in the multi-phase ACCE security game for Π.

Authentication — $\mathbf{M'(a)}$. Suppose $\mathcal{A}$ wins the weak renegotiable ACCE security experiment for Π 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 **M** of Definition 3 for Π.

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, $j \leq \ell-1$, 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 the 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.

Assuming PRF is secure and Π is a secure multi-phase ACCE, no $\mathcal{A}$ can achieve conditions $\mathbf{M'(a)}$ and **A1**–**A7**.

Authentication — $\mathbf{M'(b)}$. Now suppose $\mathcal{A}$ wins the weak renegotiable ACCE security experiment for Π 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 $\pi_A^s.\mathtt{phases}[\ell].RT_s \| RT_r \neq \pi_B^t.\mathtt{phases}[\ell].RT_r \| RT_s$. Choose ℓ minimal. Let v be the smallest index such that the vth ciphertext C_v of $\pi_A^s.\mathtt{phases}[\ell].RT_s \| RT_r$ is not equal to the vth ciphertext of $\pi_B^t.\mathtt{phases}[\ell].RT_r \| RT_s$.

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.)

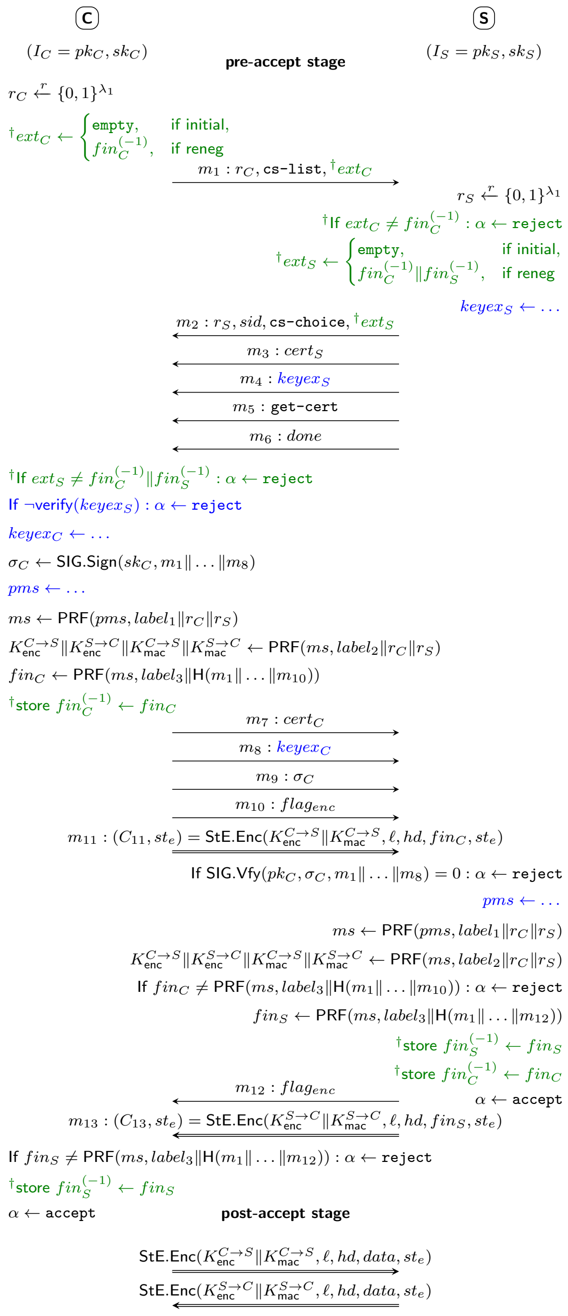
This means that when $\mathcal{A}$ called $\mathsf{Decrypt}(\pi_A^s, C_v, hd)$, if $b = 0$ then $\mathsf{Decrypt}$ returned $(\perp, \cdot)$, whereas if $b = 1$ then $\mathsf{Decrypt}$ returned $(m', \cdot)$ where $m' \neq \perp$. Our simulator can thus output (A, s, ℓ, b') for its guess of b' as above, and this will equal b with probability at least ϵ, making condition **C6** hold in Definition 3. We need to show that conditions **C1**–**C5** also hold for (A, s, ℓ).

Since $\mathcal{A}$ wins the weak renegotiable ACCE experiment using condition $\mathbf{M'(b)}$, we have that **A1**–**A7** all hold. We want to show that, at the time that π_A^s accepted in phase $\ell+1$, conditions **C1**–**C5** also hold for (A, s, ℓ).

- **C1**: **A1** implies **C1**, since if π_A^s has rejected in any phase prior to ℓ^* then it would not have a phase ℓ^*.
- **C2** and **C3**: Conditions **A2** and **A3** of Definition 5 do not imply that $\mathcal{A}$ did not ask $\mathsf{Corrupt}$ queries prohibited by **C2** and **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.
- **C4**: **A6** implies **C4**, at the time that π_A^s accepted.
- **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 **A7**, $\mathcal{A}$ 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 **C5**.

Thus, assuming Π is a secure multi-phase ACCE no $\mathcal{A}$ can achieve conditions $\mathbf{M'(b)}$ and **A1**–**A7**. $\square$

B. GENERIC TLS PROTOCOL WITH SCSV/RIE COUNTERMEASURE



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

Figure 2: Generic TLS handshake protocol with †SCSV/RIE renegotiation countermeasures