A Modular Security Analysis of the TLS Handshake Protocol

P. Morrissey, N.P. Smart and B. Warinschi

Department Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
paulm,nigel,bogdan@cs.bris.ac.uk

Abstract. We study the security of the widely deployed Secure Session Layer/Transport Layer Security (TLS) key agreement protocol. Our analysis identifies, justifies, and exploits the modularity present in the design of the protocol: the *application keys* offered to higher level applications are obtained from a *master key*, which in turn is derived, through interaction, from a *pre-master key*.

Our first contribution consists of formal models that clarify the security level enjoyed by each of these types of keys. The models that we provide fall under well established paradigms in defining execution, and security notions. We capture the realistic setting where only one of the two parties involved in the execution of the protocol (namely the server) has a certified public key, and where the same master key is used to generate multiple application keys.

The main contribution of the paper is a modular and generic proof of security for the application keys established through the TLS protocol. We show that the transformation used by TLS to derive master keys essentially transforms an *arbitrary* secure pre-master key agreement protocol into a secure master-key agreement protocol. Similarly, the transformation used to derive application keys works when applied to an arbitrary secure master-key agreement protocol. These results are in the random oracle model. The security of the overall protocol then follows from proofs of security for the basic pre-master key generation protocols employed by TLS.

1 Introduction

The SSL key agreement protocol, developed by Netscape, was made publicly available in 1994 [22] and after various improvements [20] has formed the bases for the TLS protocol [18, 19] which is nowadays ubiquitously present in secure communications over the internet. Surprisingly, despite its practical importance, this protocol had never been analyzed using the rigorous methods of modern cryptography. In this paper we offer one such analysis. Before describing our results and discussing their implications we recall the structure of the TLS protocol (Figure 1). The protocol proceeds in six phases. Through phases (1) and

(2) parties confirm their willingness to engage in the protocol, exchange, and verify the validity of their identities and public keys (it is assumed that at least one party (the server) possess a long term public/private key pair $(\mathrm{PK}_B, \mathrm{SK}_B)$, as well as a certificate $\mathrm{sig}_{\mathrm{CA}}(\mathrm{PK}_B)$ issued by some certification authority CA). The next four phases, which are the focus of this paper, are as follows.

(3) A *pre-master secret* $s \in \mathcal{S}_{\mathrm{PMS}}$ is obtained using one of a number of protocols that include RSA based key transport and signed Diffie–Hellman key exchange (which we describe and analyze later in the paper).

(4) The pre-master secret key s is used to derive a *master secret* $m \in \mathcal{S}_{\mathrm{MS}}$, with $m = G(s, r_A, r_B)$. Here r_A, r_B are random nonces that the two parties exchange and G is a key derivation function. The obtained master secret key is confirmed by using it to compute two MACs of the transcript of the conversation which are then exchanged.

(5) In the next phase the master key m is used to obtain one or more application keys: for each application key, the parties exchange random nonces n_A and n_B and compute the shared application key via $k = k' \mid\mid k'' \leftarrow H(m, n_A, n_B)$. Here, H is a key derivation function. Notice, that each application key is actually two keys: one for securing communication from the client to the server, and one from the server to the client. This is important to prevent reflection attacks.

(6) Finally the application keys are used in an application (and we exhibit one possible use for encrypting some arbitrary messages). We emphasise that many applications can use the same master key by repeated application of Steps 5 and 6.

The proper use of keys in this last stage had been the object of previous studies [4, 25] and is not part of our analysis.

An interesting aspect of TLS is that the protocols used to obtain the pre-master secret in Step (3) are very simplistic and on their own insecure in the terms of modern cryptography. It is the combination of step (3) with those in (4) and (5) which leads (as we show in this paper) to secure key agreement protocol in the standard sense. Broadly speaking, our goal is to derive sufficient security conditions on the pre-master key agreement protocol which would ensure that the above combination indeed yields a secure key-agreement protocol in a standard cryptographic sense.

We caution that in our analysis we disregard steps (1) and (2), and therefore assume an existing PKI which authenticates all public keys in use in the system. In particular we do not take into account any so-called PKI attacks.

Models. Much of the previous work on key agreement protocols in the provable security community has focused on defining security models and then creating protocols which meet the security goals of the models. In some sense, we are taking the opposite approach: we focus on a particular existing protocol, namely TLS, and develop security models that capture the security levels that the various keys derived in one execution of the protocol enjoy. The path we take is also motivated by the lack of models that capture precisely the security of these keys.

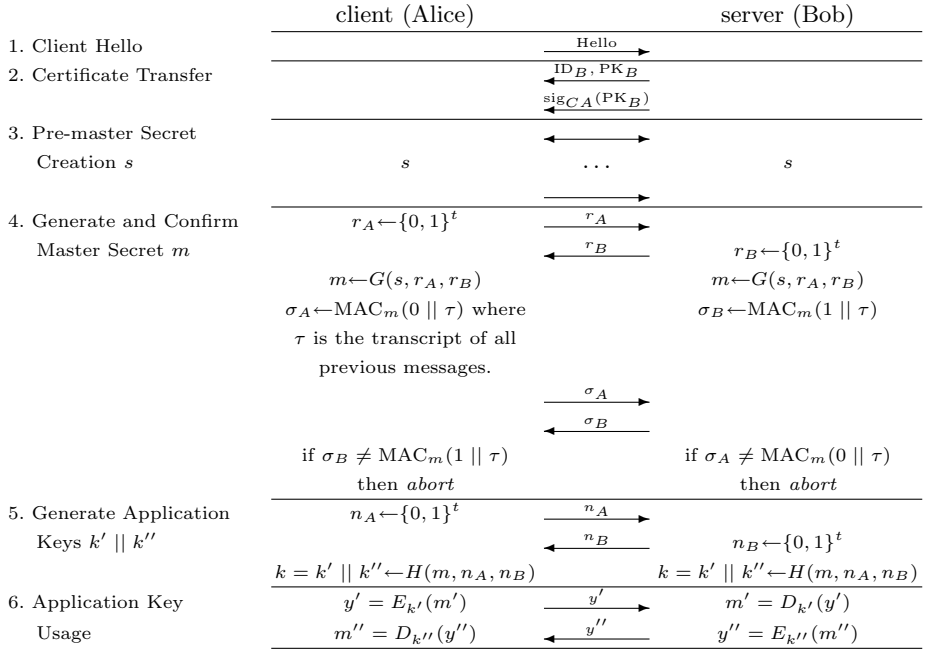

Fig. 1. A general TLS like protocol

A second important aspect of our approach is that unlike in prior work on key-agreement protocols, we do not regard the protocol as a monolithic structure. Instead, we identify the structure described above and give security models for each of the keys that are derived in the protocol. A benefit that follows from this modular approach is that we split the analysis of the overall protocol to the analysis of its components, thus making the task of proving security more manageable.

We first provide a model for pre-master key agreement protocols. The model is a weakened version of the Blake–Wilson, Johnson and Menezes (BJM) model [9]. In particular we only require that pre-master key agreement protocols are secure in a one-way sense (the adversary cannot recover the entire established key), and that the protocol is secure against man-in-the-middle attacks. In addition, unlike in prior work, we model the realistic setting where only one of the parties involved in the protocol is required to possess a certified public key.

Next, we give a security model for master-key agreement protocols which strengthens the one described above. We still only require secrecy for keys in the one-wayness sense, but now we ask for the protocol to also be secure against unknown-key-share attacks. In addition, we introduce key-confirmation as a requirement for master keys.

Finally, via a further extension, we obtain a model for the security of key agreement protocols. Our model for application key security is rather standard,

and resembles the BJM model: we require for the established key to be indistinguishable from a randomly chosen one, and we give the adversary complete control over the network, and various corruption capabilities. Our model explicitly takes into consideration the possibility that the same master key is used to derive multiple application keys.

Security analysis of the TLS handshake protocol. Based on the models that we developed, we give a security proof for the TLS handshake protocol. In particular, we analyze a version where the MAC sent in step 4 is passed in the clear (and not encrypted under the application keys as in full TLS.) It is intuitively clear that the security of the full TLS protocol follows from our analysis. While a direct analysis of the latter may be desirable we choose to trade immediate applicability of our results to full TLS for the modularity afforded by our abstraction.

Our proof is modular and generic. Specifically, we show that the protocol $(\Pi; \mathsf{MKD}_{\mathsf{SSL}}(\mathsf{Mac}, G))$ obtained by appending to an arbitrary pre-master key agreement protocol Π the flows in phase (4) of TLS is a secure master-key agreement protocol in the sense that we define in this paper. The result holds provided that the message authentication code used in the transformation is secure and the hash function in the construction is modeled as a random oracle. Similarly, we show that starting from an arbitrary secure master-key agreement protocol Π, the protocol $(\Pi; \mathsf{AK}_{\mathsf{SSL}}(H))$ obtained by appending the flows in phase (5) of TLS is a secure application-key agreement protocol (provided that H is modeled as a random oracle).

An important benefit of the modular approach that we employ surfaces at this stage: to conclude the security of the overall protocol it is sufficient to show that the individual pre-master key agreement protocols of TLS are indeed secure (in the weak sense that we put forth in this paper). The analysis is thus more manageable, and avoids duplicating and rehashing proof ideas, which would be the case if one was to analyze TLS in its entirety for each distinct method for establishing pre-master keys.

Impact on practice. An implication of practical consequence of our analysis concerns the use of encryption for implementing the pre-master key agreement protocol of TLS. Currently, the RSA key transport mode of TLS uses a randomized padding mechanism to avoid known problems with vanilla RSA. The original choice was the encryption scheme from PKCS-v1.0. The exact choice is historic, but in modern terms was made to attempt to create an IND-CCA encryption scheme. It turns out that the encryption scheme from PKCS-v1.0 is not in fact IND-CCA secure. This was exploited in the famous reaction attack by Bleichenbacher [11] on SSL, where invalid ciphertext messages were used to obtain pre-master secret keys. Our analysis implies that no randomized padding mechanism is actually needed, as deterministic encryption suffices to guarantee the security of the whole protocol.

Importantly, our models *do* capture security against reaction attacks as long as the full behaviour of the protocol is specified and analyzed. The key aspect is that the analysis should include the behaviour of the parties when the messages

that they receive do not follow the protocol (e.g. are malformed). Our analysis of the premaster key agreement based on encryption schemes (e.g. that based on RSA) considers and thus justifies the validity of the patch proposed to cope with reaction attacks, i.e. by ensuring that the execution when malformed packages are received is indistinguishable from honest executions.

Our models can be used to explicitly capture one-way and mutual authentication via public-key certificate information. We do not model variants of the standard TLS protocol which can include password-based authentication or shared key-based techniques. We leave these extensions for future work.

It is important to observe that our model does not require that the application keys satisfy a notion of key-confirmation (as we require for the master-keys). Indeed, the TLS protocol does not ensure this property. However, one may obtain implicit key confirmation through the use of such keys in further applications. In some sense, this loss is a by-product of the way we have broken up the protocol. One of our goals was to show what security properties each of the stages provides, and therefore we modeled and analyzed the security of the application keys. However, if one considers Stages 1-4 as the key agreement protocol, and stages 5-6 as the application where the keys are used, then one does obtain an explicit notion of key confirmation. Hence, the loss of explicit key confirmation in Stage 5 should not be considered a design flaw in TLS.

On the use of the random oracle model. In our proofs we assume that the key derivation function is a random oracle, i.e. an idealized randomness extractor. As such, the typical disclaimer associated to proofs in the random oracle model certainly applies, and we caution against over optimism in their interpretation. A natural and important question is whether a standard model analysis is possible, ideally, assuming that the key derivation function is pseudorandom (as is the function based on HMAC used in the current specification of TLS). Unfortunately, indirect evidence indicates that such a result is extremely hard to obtain. As observed by Jonsson and Kaliski in their analysis of the use of RSA in TLS [23], the use of the key derivation function in TLS is akin to the use of such functions in deriving DEM keys under the KEM/DEM paradigm [16]. It is thus likely that a proof as above would immediately imply an efficient RSA-based encryption scheme secure in the standard model, thus solving a long-standing open question in cryptography.

Related Work The work which is closest with ours is the analysis of the use of RSA in TLS by Jonsson and Kaliski [23]. They consider a very simplified security model for the master secret key, for the particular case when the protocol for premaster key is based on encryption. We share the modeling of the key derivation function as a random oracle, and the observation that deterministic encryption may suffice for a secure premaster key had also been made there. However, the present work uses a far more general and modular model for key-exchange, analyzes several pre-master key agreement protocols, including one based on DDH which is offered by TLS.

Other analyses of the TLS protocol used Dolev-Yao models, where ideal security of the underlying primitives is postulated, and thus no guarantees are

offered for the more concrete world. Such analyses include the one carried out by Mitchel, Shmatikov, and Stern [28] using a model checker, and the one of Paulson who used the inductive method [30]. Wagner and Schneier analyze various security aspects of SSL 3.0 [32], but their treatment is informal. Finally, Bellare and Namprempre [4], and Krawczyk [25] study how to correctly use the application keys derived via TLS. Their treatment is focused exclusively on the use of keys, and is not concerned with the security of the entire key agreement protocol.

The first complexity theoretic model for key agreement was the Bellare-Rogaway (BR) model [6, 7]. The main driving forces of this model were the works of [8, 17]. Since the initial work of Bellare and Rogaway there have been a number of other models proposed for key-exchange in various applications and environments [9, 10, 3, 13, 14, 5, 12, 1, 27, 31]. These models can be loosely categorised into two main groups: those that use simulation based techniques [3, 14, 31], and those closer to the original BR model that use an indistinguishability based approach [9, 10, 13, 27]. As explained before, our analysis uses a model that falls in the latter category which, as argued elsewhere [13], has certain drawbacks but also several important benefits over the simulation based approach. Certainly, our general understanding of TLS would benefit from an analysis in a simulation based model, especially one that guarantees compositionality [14]. However, in such settings care must be taken on the use of the UC session identifiers which must be unique and predetermined. Furthermore, multiple sessions of TLS use the same long term secret keys which is a setting inherently difficult to handle in the UC framework. The joint state UC theorem [15] a technical tool sometimes useful in such situations does not apply to encryption (as used by encryption based pre-master key derivation). Furthermore, applying the JUC theorem to protocols that use signatures it requires signing messages/session identifier pairs, thus obtaining an analysis of a related but different protocol.

Some aspects of other indistinguishability-based models relevant to our work are the following. In [6] entity authentication and authenticated key distribution are considered in the two-party symmetric key case where users are modeled as message driven oracles. The adversary in this case acts as the communications channel between users. To define security, the notions of an "error-free history" of [8] and of "matching protocol runs" from [17] are made formal in [6] using the notion of a *matching conversation*. We use this notion in our definitions.

Various security attributes are then included in the definition of security by allowing the adversary to make corresponding queries such as Reveal queries. In [7] this was developed to model the three party symmetric key case for entity authentication and key distribution. The models most relevant to our work are the Blake–Wilson, Johnson and Menezes (BJM) based models [9, 10, 27]. The BJM model of [9] extended the BR model, to authenticated key agreement (AK) and authenticated key agreement with key confirmation (AKC) in the public key case. The work of [9] uses the notion of a No-Matching condition [6], to define a clearer separation between AK and AKC protocols and deals with Diffie–Hellman

(DH) like protocols. Our execution models are inspired by the BJM model (while our security definitions are different.)

Following on from this [10] deals with the case of key transport using public key encryption (PKE) and key agreement using DH key agreement with digital signatures (DSS). In [27] a modular proof technique was used in a modified BJM model to prove security of key agreement protocols relative to a gap assumption. Indeed, the idea of transforming a one-way security definition into an indistinguishability definition occurs also in the generic transform proposed by Kudla and Paterson [26, 27] and our techniques are very similar to theirs.

Finally, an important security model that is related to ours is that of Canetti and Krawczyk (CK) [13]. In addition to the corruption capabilities that we consider, the CK model allows the adversary to obtain the entire internal state of a session and in particular the ephemeral secrets used in sessions. As pointed out by Choo et al. this type of query is the only essential difference between the adversarial capabilities in the model of Bellare and Rogaway and that of Canetti and Krawczyk (see Table 2 of [24]). Clearly, our analysis does not offer guarantees in the face of such extremely powerful types of adversaries and in fact it can be easily seen that under such attacks the TLS version that uses the DDH-based premaster secret key agreement is insecure. It may be possible that one can demonstrate security of TLS under such stronger attacks by assuming secure erasures as done for similar protocols [13, 14].

By adopting the style of the BR models over the style of the CK model we also avoid some of the idiosyncrasies of the latter related to the use of session identifiers (which need to be unique, and somehow agreed upon in advance by participating parties) [13, 24]. For a further discussion on the use of identifiers in the CK model versus the BR model see [24].

One other aspect of [13] which is somewhat related to our work is a modular framework for designing protocols. In the model of [13] one can first develop a secure protocol under the powerful assumption that all communication is authenticated. Then, a secure protocol in the more realistic setting with no authenticated communication is obtained by applying a generic transformation using an *authenticator*. Obviously, the modular structure of TLS that we observe and exploit is of a different nature. In particular it does not seem possible to regard TLS as the result of applying an authenticator to some other protocol.

Acknowledgements The authors would like to thank Caroline Belrose for various discussions on key agreement protocols during the writing of this paper and Martin Abadi for interesting insights into various aspects of TLS. The work described in this paper has been supported in part by the EU FP6 project eCrypt and an EPSRC grant.

2 A Generic Execution model for Two-Party Protocols

The security models that we use in this paper are based on the earlier work of Bellare et al. [3, 5–7], as refined by BJM [9]. In this section we give a general

description of the common features of these models, and recall some of the intuition behind them. Later, we specialise the general model for the different tasks that we consider in the paper.

Registered and unregistered users. We model a setting with two kinds of users: registered users (with identities in some set $\mathcal{U}$) and non-registered user (with identities in some set $\mathcal{U}'$). Each user $U \in \mathcal{U}$ has a long-term public key PK_U and a corresponding long term private key SK_U. The set $\mathcal{U}$ is intended to model the set of servers in the standard one-way authentication mode of TLS, the set of identities $\mathcal{U}'$ models users that do not have a long term public/private key pair.

Models for interactive protocols execution. We are concerned with two-party protocols: interactive programs in which an initiator and a responder communicate via some communication channel. Each of the two parties runs some reactive program: each program expects to receive a message from the communication channel, computes a response, and sends this back to the channel. We refer to one execution of the program for the initiator (respectively, responder) as an initiator session (respectively, a responder session). Each party may engage in multiple, concurrent, initiator and responder sessions.

As standard, we assume an adversary in absolute control of the communication network: the adversary intercepts all messages sent by parties, and may respond with whatever message it wants. This situation is captured by considering an adversary (an arbitrary probabilistic, polynomial-time algorithm) who has access to oracles that correspond to some (initiator or responder) sessions of the protocol which the oracle maintains internally. In particular, each oracle maintains an internal state which consists of the variables of the session to which it corresponds, and additional meta-variables used later to define security notions. In our descriptions we typically ignore the details of the local variables of the sessions, and we omit a precise specification of how these sessions are executed. Both notions are standard. The typical meta-variables of an oracle $\mathcal{O}$ include the following. Variable $\tau_{\mathcal{O}} \in \{0,1\}^* \cup \{\perp\}$ that maintains the transcript of all messages sent and received by the oracle, and occasionally, other data pertaining to the execution. Variable $\mathrm{role}_{\mathcal{O}} \in \{\mathit{initiator}, \mathit{responder}, \perp\}$ records the type of session to which the oracle corresponds. Variable $\mathrm{pid}_{\mathcal{O}} \in \mathcal{U}$ keeps track of the identity of the intended partner of the session maintained by $\mathcal{O}$. Variable $\delta_{\mathcal{O}}$ indicates whether the session had finished successfully, or unsuccessfully. We specify the values that this variable takes later in the paper. Finally, variable $\gamma_{\mathcal{O}} \in \{\perp, \mathit{corrupted}\}$ records whether or not the session had been corrupted by the adversary.

After an initialisation phase, in which long term keys for the parties are generated the adversary takes control of the execution which he drives forward using several types of queries. The adversary can create a new session of user U playing the role of the initiator/responder by issuing a query $\mathsf{NewSession}(U, \mathit{role})$, with $\mathit{role} \in \{\mathit{initiator}, \mathit{responder}\}$. User U can be either registered or unregistered. We write Π_U^i for the i'th session of user U. To any oracle $\mathcal{O}$ the adversary can send a message msg using the query $\mathsf{Send}(\mathcal{O}, \mathrm{msg})$. In return the adversary

receives an answer computed according to the session maintained by $\mathcal{O}$. The adversary may also corrupt oracles. Later in the paper when we specialise the general model, we also clarify the different versions of corruptions that can occur and how are they handled by the oracles. The execution halts whenever the adversary decides to do so.

To identify sessions that interact with each other we use the notion of matching conversations introduced by Bellare and Rogaway (which essentially states that the inputs to one session are outputs of the other sessions, and the other way around) [6].

3 Pre-Master Key Agreement Protocols

In this section we specialise the general model described above for the case of pre-master key agreement protocols, and analyze the security of the pre-master key agreement protocols used in TLS.

As discussed in the introduction, the design of our models is guided by the security properties that the various subprotocols of TLS satisfy. In particular, we require extremely weak security properties for the pre-master secret key. Specifically, we demand that an adversary is not able to *fully* recover the key shared between two honest parties. In its attack the adversary is allowed to adaptively corrupt parties and obtain their long term secret key, and is allowed to check if a certain string s equals the pre-master secret key held by some honest session. The latter capability models an extremely limited form of reveal queries: our adversary is not allowed to obtain the pre-master secret key of any of the sessions, but can only guess (and then check) their values.

The formal model of security for pre-master key agreement protocols extends the general model in Section 2 and makes only mild assumptions regarding the syntax of such protocols. Specifically, we assume that the pre-master key belongs to some space $\mathcal{S}_{\mathrm{PMS}}$. This space is often the support set of some mathematical structure such as a group. We require that if t is the security parameter then $\#\mathcal{S}_{\mathrm{PMS}} \geq 2^t$. Furthermore, we assume that the initiator and responder programs use a variable $s \in \mathcal{S}_{\mathrm{PMS}} \cup \{\perp\}$ that stores the shared pre-master key. The corresponding variable stored by some oracle $\mathcal{O}$ is $s_{\mathcal{O}}$. For pre-master secret key agreement protocols the internal variable $\delta_{\mathcal{O}}$ stores one of the following values: $\perp$ (the session had not finished its execution), *accepted-pmk* (the session had finished its execution successfully (which in particular means that $s_{\mathcal{O}}$ holds some pre-master session key in $\mathcal{S}_{\mathrm{PMS}}$) or *rejected* (the session had finished its execution unsuccessfully). Unless $\delta_{\mathcal{O}} =$ *accepted-pmk* we assume $s_{\mathcal{O}} = \perp$.

The corruption capabilities of the adversary discussed above are modeled using queries $\mathsf{Corrupt}$ and Check formally defined as follows. When the adversary issues a query $\mathsf{Corrupt}(U)$ the following actions take place. If $U \in \mathcal{U}$ then $\mathrm{SK}_{\mathcal{U}}$ is returned to the adversary, and we say that party U had been corrupted. In all sessions $\mathcal{O} = \Pi^i_U$ for some $i \in \mathbb{N}$ the value of $\gamma_{\mathcal{O}}$ is set to *corrupted* and no further interaction between these oracles and the adversary may take place. Additionally, no further queries $\mathsf{NewSession}(U, role)$ are permitted.

When the adversary issues the query $\mathsf{Check}(\mathcal{O}, s)$, for $\mathcal{O} = \Pi_U^i$, $i \in \mathbb{N}$, U some uncorrupted party, and $s \in \mathcal{S}_{\mathrm{PMS}}$, then the answer returned to the adversary is **true**, if $\delta_{\mathcal{O}} = \textit{accepted-pmk}$ and $s_{\mathcal{O}} = s$, and **false** otherwise. When a given oracle is initialized all values for the internal states are set to $\perp$. At the end of a protocol, the role, partner ID, and oracle state (but not the pre-master key) are recorded in the transcript.

The following definition captures the class of oracles which are valid targets for the attacker using the notion of "fresh oracles". These are uncorrupted oracles who have successfully finished their execution, and have a known intended partner who is also not corrupted.

Definition 1 (Fresh Pre-Master Secret Key Oracle). *A pre-master secret oracle $\mathcal{O}$ is said to be fresh if all of the following conditions are satisfied:*

(1) $\gamma_{\mathcal{O}} = \perp$, (2) $\delta_{\mathcal{O}} = \textit{accepted-pmk}$, and (3) $\exists\ V \in \mathcal{U}$ such that V is uncorrupted and $\mathrm{pid}_{\mathcal{O}} = V$.

Security game for pre-master key agreement protocols. We define the security of a pre-master key agreement protocol Π via the following game $\mathsf{Exec}^{\mathsf{OW\text{-}PMS}}_{\mathcal{A},\Pi}(t)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

(1) The challenger, $\mathcal{C}$, generates public/secret key pairs for each user $U \in \mathcal{U}$ (by running the appropriate key-generation algorithm on the security parameter t), and returns the public keys to $\mathcal{A}$.

(2) Adversary $\mathcal{A}$, is allowed to make as many $\mathsf{NewSession}$, Send, Check, and $\mathsf{Corrupt}$ queries as it likes.

(3) At some point $\mathcal{A}$ outputs a pair $(\mathcal{O}^*, s^*)$, where $\mathcal{O}^*$ is some pre-master secret oracle, and $s^* \in \mathcal{S}_{\mathrm{PMS}}$.

We say the adversary $\mathcal{A}$ wins if its output $(\mathcal{O}^*, s^*)$ is such that $\mathcal{O}^*$ is fresh, and $s^* = s_{\mathcal{O}^*}$. In this case the output of $\mathsf{Exec}^{\mathsf{OW\text{-}PMS}}_{\Pi,\mathcal{A}}(t)$ is set to 1. Otherwise the output of the experiment is set to 0. We write

$$\mathbf{Adv}^{\mathsf{OW\text{-}PMS}}_{\mathcal{A},\Pi}(t) = \Pr[\mathsf{Exec}^{\mathsf{OW\text{-}PMS}}_{\mathcal{A},\Pi}(t) = 1],$$

for the advantage of $\mathcal{A}$ in winning the $\mathsf{Exec}^{\mathsf{OW\text{-}PMS}}_{\mathcal{A},\Pi}(t)$ game. The probability is taken over all the random coins used in the game. We deem a pre-master secret key protocol secure if the adversary is not able to fully compute the key held by fresh oracles.

Definition 2 (Pre-Master Key Agreement Security). *A pre-master key agreement protocol is secure if it satisfies the following requirements:*

- **Correctness:** *If at the end of the execution of a benign adversary, who correctly relays messages, any two oracles which have had a matching conversation hold the same pre-master key, and the key should be distributed uniformly on the pre-master key space $\mathcal{S}_{\mathrm{PMS}}$.*
- **Key Secrecy:** *A pre-master key agreement protocol Π satisfies* OW-PMS *key secrecy if for any* p.p.t. *adversary $\mathcal{A}$ its advantage $\mathbf{Adv}^{\mathsf{OW\text{-}PMS}}_{\mathcal{A},\Pi}(t)$ is a negligible function.*

Before proceeding, we discuss the strength of our model for the security of pre-master secret keys, and several authentication issues.

REMARK 1. Our security requirements for pre-master secret key agreement are significantly weaker than the standard requirements for key exchange [6, 7]. In particular, we only require secrecy in the sense of one-wayness (not in the sense of indistinguishability from a random key). Furthermore, the corruption abilities of the adversary are severely limited: the adversary cannot obtain (or "reveal") pre-master secrets established by honest parties (even if these parties are not those under the attack).

REMARK 2. As a consequence of our security requirements our model may deem protocols that succumb to unknown-key-share attacks [17] secure. In such attacks, two sessions belonging to honest users U and V locally establish the same pre-master secret key, without intentional interaction with each other.

REMARK 3. Security under our notion guarantees security against man-in-the-middle attacks: a situation where honest parties U and V believe they interact with each other but their pre-master key(s) is in fact shared with the adversary is a security break in our model.

REMARK 4. Although the resulting security notion is very weak, it turns out that it suffices to obtain good master-key agreement protocols by appropriately designed protocols to derive such keys (e.g. the protocol in Step 4 of the TLS protocol – Figure 1.) More importantly, the weak notion also allows for many simple protocols to be proved secure. For example, in the next section we prove that deterministic encryption is sufficient to construct such protocols.

REMARK 5. Our model is not concerned with secure establishment of pre-master secret keys between two unauthenticated parties (the oracle that is under attack always has $\mathrm{pid}_{\mathcal{O}} \neq \perp$). While treating this case is possible using the concept of matching conversations to pair sessions, the resulting definition would be heavier and not particularly illuminating. Instead, we concentrate on the situation more relevant to practice where at least one of the parties that take part in the protocol (the server) has a certified public key.

REMARK 6. As usual, our security model can be easily adapted to the random oracle model by providing the adversary with access to the random oracle (whenever some hash function is modeled as a RO). The same holds true for the rest of the models that we develop in this paper.

We now discuss the security of the pre-master secret key agreement protocols used in TLS.

PROTOCOLS BASED ON PUBLIC-KEY ENCRYPTION. A natural, intuitively appealing, construction for pre-master key agreement protocols is based on the following use of an arbitrary public-key encryption scheme Enc. A user selects a pre-master secret key s from an appropriate space, and sends to the server the encryption of s under the server's public-key. The server then obtains s as the decryption of the ciphertext that it receives. We write $\mathsf{PMK}(\mathsf{Enc})$ for the resulting protocol.

Theorem 1. *If* Enc *is a* OW-CPA *secure deterministic encryption or a* OW-CCA *secure randomized encryption scheme, then the pre-master secret key agreement protocol* $\Pi = \mathsf{PMK}(\mathsf{Enc})$ *is a secure pre-master key transport protocol.*

The result of this theorem, like all theorems in this paper will be proved in the full version.

The weak security properties that we define for pre-master key agreement protocols enable us to show security of PMK(Enc) based on weak security requirements for Enc. Indeed, the one-wayness type secrecy for pre-master keys translates to the one-wayness of the encryption function of Enc. This result of our analysis implies, perhaps surprisingly, that one can avoid the use of full-fledged IND-CCA encryption schemes in favor of the much simpler *deterministic* OW-CPA schemes (*e.g.* textbook RSA). Of course, probabilistic encryption can also be used, but in this case we show security of the associated pre-master secret key protocol based on OW-CCA security. More generally our results holds under the assumption that the encryption scheme is secure against an attacker with access to a plaintext checking oracle. It is therefore not paradoxical that a deterministic scheme suffices but an IND-CPA scheme does not.

Finally, since IND-CCA implies OW-CCA, our security analysis *does* apply to the (correct) use of an IND-CCA secure public key encryption scheme within the TLS protocol. In particular, when Enc is RSA-OAEP, the pre-master secret key protocol PMK(Enc) is secure.

Signed Diffie-Hellman pre-master key agreement. The pre-master secret key in TLS can also be produced by exchanging a Diffie-Hellman key g^{xy}, for x and y randomly chosen by the two participants, who also sign the relevant message flow (either g^x or g^y) with their long term signing keys. It is known that this protocol, which we denote by $\mathsf{PMK}(\mathsf{Sig}, \mathbb{G})$, does not meet the requirements of an authenticated key agreement protocol, for example see [17] for a discussion of this protocol and various attacks on it. However, one can show

Theorem 2. *Let* $\mathbb{G}$ *be cyclic group for which the gap-Diffie-Hellman assumption holds and let* Sig *be a secure digital signature scheme. Then* $\Pi = \mathsf{PMK}(\mathsf{Sig}, \mathbb{G})$ *is a secure pre-master key agreement protocol.*

4 Master Key Agreement Protocols

In this section we introduce a security model for master-key agreement protocols. We then show that master key agreement protocols obtained from secure pre-master key agreement protocols via the transformation used in TLS satisfy our notion of security.

Our security model for master key agreement protocols is similar to that for pre-master key agreement protocols. We again ask for the adversary not to be able to *fully* recover the master secret key of the session under attack. Moreover, we ask for a key confirmation guarantee: if a session of some user U accepts a certain master-key then there exists a unique session of its intended partner that had accepted the same key. In addition to the queries previously

defined for the adversary, we also let the adversary obtain the master keys agreed in different sessions of the protocol, without corrupting the user to which this session belongs, i.e. we allow so-called Reveal queries.

In the formal model that we give below we make the following assumptions about the syntax of a master-key agreement protocol. We assume that the master key belongs to some space $\mathcal{S}_{\text{MS}}$ for which we require that $\#\mathcal{S}_{\text{MS}} \geq 2^t$, and assume that the programs that specify a master key agreement protocol use a variable m to store the agreed master key. For such protocols the variable $\delta_{\mathcal{O}}$ now takes values in $\{\perp, \textit{accepted-mk}, \textit{reject}\}$ with the obvious meaning. Furthermore, the variable $\gamma_{\mathcal{O}}$ can also take the value *revealed* to indicate that the stored master key has been given to the adversary (see below).

In addition to the queries allowed in the experiment for pre-master key security, the adversary is also allowed to issue queries of the form $\mathsf{Reveal}(\mathcal{O})$. This query is handled as follows: if $\delta_{\mathcal{O}} = \textit{accepted-mk}$ then $m_{\mathcal{O}}$ is returned to $\mathcal{A}$ and $\gamma_{\mathcal{O}}$ is set to *revealed*, while if $\delta_{\mathcal{O}} \neq \textit{accepted-mk}$ then the query acts as a no-op. As before, when a given oracle is initialized all values for the internal states are set to $\perp$. At the end of a protocol the role, partner ID and oracle state (but not the master key) are recorded in the transcript. Unless $\delta_{\mathcal{O}} = \textit{accepted-mk}$ we assume $m_U^i = \perp$.

The definition of freshness needs to be adapted to take into account the new adversarial capabilities. We call an oracle $\mathcal{O}$ fresh if it is uncorrupted, has successfully finished its execution, its intended partner V is uncorrupted, and none of the revealed oracles belonging to V has had a matching conversation with $\mathcal{O}$. The latter condition essentially says that the adversary can issue $\mathsf{Reveal}(\mathcal{Q})$ for any $\mathcal{Q}$ (including those that belong to the intended partner of $\mathcal{O}$), as long as $\mathcal{Q}$ is not the session with which $\mathcal{O}$ actually interacts.

Definition 3 (Fresh Master Secret Oracle). *A master secret oracle $\mathcal{O}$ is said to be fresh if all of the following conditions hold:*

(1) $\gamma_{\mathcal{O}} = \perp$, *(2)* $\delta_{\mathcal{O}} = \textit{accepted-mk}$, *(3)* $\exists\ V \in \mathcal{U}$ *such that V is uncorrupted and* $\text{pid}_{\mathcal{O}} = V$, *and*

(4) No revealed oracle Π_V^i has had a matching conversation with $\mathcal{O}$.

Security game for master-key agreement protocols. The game, denoted by $\mathsf{Exec}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t)$, for defining the security of master-key agreement protocol Π in the presence of adversary $\mathcal{A}$ is similar to that for pre-master key, with the modification that $\mathcal{A}$ is also allowed to make any number of Reveal queries, in addition to the $\mathsf{NewSession}$, Send, $\mathsf{Corrupt}$, Reveal, and Check queries. Here, check queries are with respect to the master secret keys only. When the adversary stops, it outputs a pair $(\mathcal{O}^*, m^*)$, where $\mathcal{O}^*$ identifies one of its oracles, and m^* is some element of $\mathcal{S}_{\text{MS}}$. We say that $\mathcal{A}$ wins if its output $(\mathcal{O}^*, m^*)$ is such that O^* is fresh and $m^* = m_{\mathcal{O}^*}$. In this case the output of $\mathsf{Exec}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t)$ is set to 1. Otherwise the output of the experiment is set to 0. We write

$$\mathbf{Adv}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t) = \Pr[\mathsf{Exec}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t) = 1]$$

for the advantage of $\mathcal{A}$ in winning the $\mathsf{Exec}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t)$ game. The probability is taken over all random coins used in the execution.

The following definition describes a situation where some party U had engaged in a session which terminated successfully with some party V, but no session of V has a matching conversation with U.

Definition 4 (No-Matching). *Let* $\mathsf{No\text{-}Matching}_{\mathcal{A},\Pi}(t)$ *be the event that at some point during the execution of* $\mathsf{Exec}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t)$ *for two uncorrupted parties* $U \in \mathcal{U} \cup \mathcal{U}'$ *and* $V \in \mathcal{U}$ *there exists an oracle* $\mathcal{O} = \Pi^i_U$ *with* $\mathrm{pid}_{\mathcal{O}} = V \in \mathcal{U}$, $\delta_{\mathcal{O}} =$ *accepted, and yet no oracle* Π^i_V *has had a matching conversation with* $\mathcal{O}$.

The following definition says that a protocol is a secure master-key agreement protocol if the key established in an honest session is secret (in the one-wayness sense) and no honest party can be coaxed into incorrectly accepting.

Definition 5 (Master Key Agreement Security). *A master key agreement protocol is secure if it satisfies the following requirements:*

- **Correctness:** *If at the end of the execution of a benign adversary, who correctly relays messages, any two oracles which have had a matching conversation hold the same master key, which is distributed uniformly over the master key space* $\mathcal{S}_{\mathrm{MS}}$.
- **Key Secrecy:** *A master key agreement protocol* Π *satisfies* OW-MS *key secrecy if for any* p.p.t. *adversary* $\mathcal{A}$, *its advantage* $\mathbf{Adv}^{\mathsf{OW\text{-}MS}}_{\mathcal{A},\Pi}(t)$ *is a negligible function.*
- **No Matching:** *For any* p.p.t. *adversary* $\mathcal{A}$, *the probability of the event* $\mathsf{No\text{-}Matching}_{\mathcal{A},\Pi}(t)$ *is a negligible function.*

Remark 1. Our security requirements for master secret keys are still significantly weaker than the more standard requirements for key exchange [6, 7]. Although the adversarial powers are similar to those in existing models (*e.g.*[9]), we still require the adversary to recover the entire key. The weaker requirement is motivated by our use of TLS as guide in designing the security model. In this protocol, the master secret key is *not* indistinguishable from a random one since it is used to compute MACs that are sent over the network.

Remark 2. The No Matching property we require is essentially the one based on matching conversations introduced by Bellare and Rogaway [6], adapted to our setting where only one of the parties involved in the execution is required to hold a certified key (and thus have a verifiable identity). One could potentially replace matching conversations with weaker versions of partnering, but only at the expense of making the definitions and results less clear. Bellare and Rogaway also show that if the No Matching property is satisfied, then agreement is injective. In our terms, with overwhelming probability it holds that if $\mathcal{O} = \Pi^i_U$ had accepted and has $\mathrm{pid}_{\mathcal{O}} = V \in \mathcal{U}$, then there exist precisely one session of V with which $\mathcal{O}$ has a matching conversation.

Remark 3. Notice that, together, the first and third conditions in the above definitions imply a key confirmation guarantee: if one session has accepted a certain key, then there exists a unique session of the intended partner who has accepted the same key.

Remark 4. The addition of Reveal queries implies security against "unknown-key-share" attacks: if parties U and V share a master-key without being aware that they interact with each other the adversary can obtain the key of U by performing a Reveal query on the appropriate session of V, thus breaking security in the sense defined above.

Remark 5. Notice that an adversary against the master-secret key does not have any query that allows it to obtain information about the pre-master secret key. This is consistent with the SSL specification which states that the pre-master secret should be converted to the master secret immediately and that the pre-master secret should be securely erased from memory. In particular this means that the pre-master secret does not form part of the state of the master key agreement oracle, and so it does not get written on a transcript.

In this section we show that the master-key agreement protocol obtained from a secure pre-master key agreement protocol by using the transformation used in TLS is secure. Let Π be an arbitrary pre-master key agreement protocol, G a hash function, and $\mathsf{Mac} = (\mathcal{K}, \mathrm{MAC}, \mathrm{ver})$ a message authentication code. We write $(\Pi; \mathsf{MKD}_{\mathsf{SSL}}(\mathsf{Mac}, G))$ the master-key agreement protocol obtained by extending Π with the master-key derivation phase of TLS, i.e. by appending to the message flows of Π those in Step 4 of Figure 1. Starting from a secure pre-master key agreement protocol, the above transformation yields a secure master key agreement protocol.

Theorem 3. *Let Π be a secure pre-master agreement protocol,* Mac *a secure message authentication code, and G a random oracle. Then $(\Pi; \mathsf{MKD}_{\mathsf{SSL}}(\mathsf{Mac}, G))$ is a secure master-key agreement protocol.*

5 Application Key Agreement

In this section we extend the model developed so far to deal with application keys obtained from master-secret keys, and the analyze the security of the application keys obtained through the TLS protocol.

As discussed in the introduction we focus on protocols with a particular structure: first, a master-key is agreed by the parties via some master-key agreement protocol Π, and then this key is used as input to an application key derivation protocol, Σ. The same master-key can be used in multiple executions of the application key protocol which can take place in parallel and concurrently.

We capture this setting by modifying the model for master-key agreement protocols as follows. We consider two types of oracles: MK-oracles which correspond to sessions where the master secret key is derived (i.e. sessions of protocol Π), and AK-oracles, which correspond to sessions of the application key derivation protocol (i.e. sessions of Σ). The AK-oracles are spawned by MK-oracles that have established a master-secret key; spawning is done at the request of the adversary. The internal structure and behavior of MK-oracles are as defined in the previous section. To describe AK-oracles, we again impose some syntactic

restrictions on the protocols (and thus on the oracles). We require that AK-oracle $\mathcal{Q}$ maintain variables $\tau_{\mathcal{Q}}, m_{\mathcal{Q}}, \mathrm{role}_{\mathcal{Q}}, \mathrm{pid}_{\mathcal{Q}}$ with the same roles as before. In addition, a new variable $k_Q \in \mathcal{S}_{\mathrm{A}}$ holds the application key obtained in the session. (Here $\#\mathcal{S}_{\mathrm{A}} \geq 2^t$, where t is the security parameter). The state variable $\delta_{\mathcal{Q}}$ now assumes values in $\{\perp, \textit{accepted-ak}, \textit{rejected}\}$, with the obvious semantics. Finally, the corruption variable $\delta_{\mathcal{Q}}$ is either $\perp$ or *compromised* (we explain below when the latter value is set).

In addition to the powers previously granted to the adversary, now the adversary can also create new AK-oracles by issuing queries of the form $\mathsf{Spawn}(\mathcal{O})$, with $\mathcal{O}$ an MK-oracle that had successfully finished its execution. As a result, a new oracle $\mathcal{Q} = \Sigma^j_{\mathcal{O}}$ is created (where j indicates that $\mathcal{Q}$ is the j'th oracle spawned by $\mathcal{O}$.) Oracle $\mathcal{Q}$ inherits the variables $\tau_{\mathcal{Q}}$, $m_{\mathcal{Q}}$, $\mathrm{role}_{\mathcal{Q}}$, and $\mathrm{pid}_{\mathcal{Q}}$ from $\mathcal{O}$ in the obvious way. The adversary may also compromise AK-oracles: when a query $\mathsf{Compromise}(\mathcal{Q})$ is issued, if $\mathcal{Q}$ has accepted, then $k_{\mathcal{Q}}$ is returned to the adversary and $\delta_{\mathcal{Q}}$ is set to *compromised*. Notice that the Compromise queries are the analogue of Reveal queries for AK-oracles. We chose to have different names for clarity.

The security of keys is captured via a Test query. When $\mathsf{Test}(\mathcal{Q})$ is issued, a bit $b \in \{0, 1\}$ is chosen at random. Then if $b = 0$ then $k_{\mathcal{Q}^*}$ is returned to the adversary, otherwise a randomly selected element from $\mathcal{S}_{\mathrm{A}}$ is returned to the adversary (who then has to guess b; see the game defined below).

An AK-oracle $\mathcal{Q}$ is a valid target for the adversary if the parent oracle of $\mathcal{Q}$ is fresh, $\mathcal{Q}$ has finished successfully its execution, its intended partner, say V, is not corrupt, and any session of V with which $\mathcal{Q}$ has a matching conversation is not compromised.

Definition 6 (Fresh Application Key Oracle). *Let $\mathcal{O}$ be a master key agreement oracle and $\mathcal{Q}$ denote one of its children. The oracle $\mathcal{Q}$ is said to be fresh if the following conditions hold:*

(1) $\mathcal{O}$ is a fresh master key agreement oracle, (2) $\gamma_{\mathcal{Q}} = \perp$, (3) $\delta_O =$ accepted-ak , (4) $\exists\, V \in \mathcal{U}$ such that $\mathrm{pid}_{\mathcal{Q}} = V$, and (5) No compromised session $\Sigma_{\mathcal{Q}'}$ that belongs to V has had a matching conversation with $\mathcal{Q}$.

Note that here, we are implicitly assuming that knowing a master key automatically gives the adversary all derived application keys. Whilst this will not be true of all protocols which one can think of, it is true for all application key derivation protocols that we consider here and in particular in Stage 5 of the protocol of Figure 1.

Security game for application-key agreement protocols. We define the security of an application-key protocol $\Pi; \Sigma$ via a game $\mathsf{Exec}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},\Pi;\Sigma}(t)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

(1) $\mathcal{C}$ generates public-secret key pairs for each user $U \in \mathcal{U}$, and returns the public keys to $\mathcal{A}$.

(2) $\mathcal{A}$ is allowed to make as many NewSession, Send, Spawn, Compromise, Reveal, Check, and Corrupt queries as it likes throughout the game.

(3) At any point during the game adversary $\mathcal{A}$ makes a single $\mathsf{Test}(\mathcal{Q}^*)$ query.

(4) The adversary outputs a bit b'.

We say that $\mathcal{A}$ wins if $\mathcal{Q}^*$ is fresh at the end of the game and its output bit b is such that $b = b'$ (where b is the bit internally selected during the Test query). In this case the result of $\mathsf{Exec}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},\Pi;\Sigma}(t)$ is set to 1. Otherwise the output of the experiment is set to 0. We write

$$\mathbf{Adv}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},(\Pi;\Sigma)}(t) = \left|\Pr[\mathsf{Exec}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},\Pi;\Sigma}(t) = 1] - \frac{1}{2}\right|$$

for the advantage of $\mathcal{A}$ in winning the $\mathsf{Exec}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},\Pi;\Sigma}(t)$ game. Using this security game we can now define the security of a application key agreement protocol.

Definition 7 (Application Key Agreement Security). *An application key agreement protocol is secure if it satisfies the following conditions:*

- **Correctness:** *In the presence of an adversary which faithfully relays messages, two oracles running the protocol accept holding the same application key and session ID, and the application key is distributed uniformly at random on the application key space.*
- **Key secrecy:** *An application key agreement protocol $\Pi;\Sigma$ satisfies* IND-AK *key secrecy if for any* p.p.t. *adversary $\mathcal{A}$, its advantage $\mathbf{Adv}^{\mathsf{IND\text{-}AK}}_{\mathcal{A},\Pi;\Sigma}(t)$ is negligible in t.*

Remark 1. The model that we develop ensures strong security guarantees for the application keys, in the standard sense of indistinguishability against attackers with powerful corruption capabilities. In this sense our model is close to existing ones, but has the added feature that we explicitly consider the setting where more than one application-key can be derived from the same master key.

Remark 2. Notice that at the application key layer we do not require key confirmation anymore. Indeed, a trivial attack on the standard notion of key confirmation can be mounted against application keys derived using the TLS protocol. However, implicit key confirmation for application keys may still be achieved, depending how the application key is actually used. (In the full version of the paper we discuss the composition of our application key agreement protocol with specific applications, especially confidentiality applications.)

The loss of this property is in some sense a result of how we chose to break down the protocol for analysis, since one of our goals was to identify what security properties each of the stages provides. However, if one considers Stages 1-4 as the key agreement protocol, and stages 5-6 as the application then one does obtain an explicit notion of key confirmation. Hence, the loss of explicit key confirmation in Stage 5 should not be considered a design flaw in TLS.

In this section we show that the application-key agreement protocol obtained from any secure master-key derivation protocol, and the application-key derivation protocol of TLS (Stage 5 of Figure 1) is secure.

For any master-key agreement protocol Π, and hash function H, we write $(\Pi;\mathsf{AK}_{\mathsf{SSL}}(H))$ for the application-key agreement protocol obtained by extending

Π with the application-key derivation protocol of TLS. Informally, this means that we derive an application key agreement protocol from a master key agreement protocol using Stage 5 of Figure 1. We make no assumption as to whether the master key agreement protocol itself is derived from a pre-master key agreement protocol as in Figure 1. The following theorem says that starting with a master-key agreement protocol secure in the sense of Definition 5, the above transformation yields a secure application key protocol

Theorem 4. *Let Π be a secure master-key agreement protocol and H a random oracle. Then $(\Pi; \mathsf{AK}_{\mathsf{SSL}}(H))$ is a secure application-key agreement protocol.*

The security of TLS follows from Theorems 1, 2, 3 and 4. For full details the reader should consult the full version of this paper.

References

1. M. Abdalla and O. Chevassut and D. Pointcheval. One–Time Verifier–based Encrypted Key Exchange. In *Public Key Cryptography – PKC 2005* Springer-Verlag LNCS 3386, 47–64, 2005.
2. J.H. An, Y. Dodis and T. Rabin. On the Security of Joint Signature and Encryption. In *EUROCRYPT 2002*, Springer-Verlag LNCS 2332, 83–107, 2002.
3. M. Bellare, R. Canetti and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In 30th Symposium on Theory of Computing – STOC 1998, ACM, 419–428, 1998.
4. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT 2000*, Springer-Verlag LNCS 1976, 531–545, 2000.
5. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 139–155, 2000.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO '93*, Springer-Verlag LNCS 773, 232–249, 1994.
7. M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In 27th Symposium on Theory of Computing – STOC 1995, ACM, 57–66, 1995.
8. R. Bird, I.S. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva and M. Yung Systematic Design of Two-Party Authentication Protocols. In *Advances in Cryptology – CRYPTO '91* Springer-Verlag LNCS 576, 44–61, 1991.
9. S. Blake–Wilson, D. Johnson and A.J. Menezes. Key agreement protocols and their security analysis. In *Cryptography and Coding*, Springer-Verlag LNCS 1355, 30–45, 1997.
10. S. Blake–Wilson and A. Menezes. Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques. In *IWSP*, Springer-Verlag LNCS 1361, 137–158, 1998.
11. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology – CRYPTO '98*, Springer-Verlag LNCS 1462, 1–12, 1998.

12. E. Bresson, O. Chevassut and D. Pointcheval. Provably Authenticated Group Diffie–Hellman Key Exchange – The Dynamic Case. In *Advances in Cryptology – ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 290–309, 2001.
13. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology – EUROCRYPT 2001*, Springer-Verlag LNCS 2045, 453-474, 2001.
14. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Advances in Cryptology – EUROCRYPT 2002*, Springer-Verlag LNCS 2332, 337–351, 2002.
15. R. Canetti and T. Rabin. Universal Composition with Joint State, In *Advances in Cryptology – CRYPTO 2003*, Springer-Verlag LNCS 2729, 265–281, 2003.
16. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, **33**, 167–226, 2003.
17. W. Diffie, P.C. van Oorschot and M.J. Weiner. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, **2**, 107–125, 1992.
18. T. Dierks and C. Allen. *The TLS Protocol Version 1.0.* RFC 2246, January 1999.
19. T. Dierks and C. Allen. *The TLS Protocol Version 1.2.* RFC 4346, April 2006.
20. A.O. Freier, P. Karlton and P.C. Kocher. *The SSL Protocol Version 3.0.* Internet Draft, 1996.
21. P. Fouque, D. Pointcheval and S. Zimmer. HMAC is a Randomness Extractor and Applications to TLS. *Symposium on Information, Computer and Communications Security (ASIACCS'08)*
22. K. E. B. Hickman. *The SSL Protocol Version 2.0.* Internet Draft, 1994.
23. J. Jonsson and B. Kaliski Jr. *On the Security of RSA Encryption in TLS* In *Advances in Cryptology – CRYPTO 2002*, Springer-Verlag, LNCS 2442, 127-142, 2002.
24. K.-K. R. Choo, C. Boyd and Y. Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In *Advances in Cryptology – ASIACRYPT 2005*, Springer-Verlag, LNCS 3788, 585–604, 2005.
25. H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology – CRYPTO 2001*, Springer-Verlag LNCS 2139, 310–331, 2001.
26. C. Kudla. *Special signature schemes and key agreement protocols.* PhD Thesis, Royal Holloway University of London, 2006.
27. C. Kudla and K. Paterson. Modular security proofs for key agreement protocols. In *Advances in Cryptology – ASIACRYPT 2005*, Springer-Verlag LNCS 3788, 549–565, 2005.
28. J.C. Mitchell, V. Shmatikov and U. Stern. Finite-state analysis of SSL 3.0. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998*, 1998.
29. L. Mazare and B. Warinschi. On the security of encryption under adaptive corruptions. Preprint, 2007.
30. L. Paulson. Inductive analysis of the Internet protocol TLS. In *ACM Transations on Information and Systems Security*, 2(3):332–351, 1999.
31. V. Shoup. On formal models for secure key exchange (version 4). Preprint, 1999.
32. D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *"2nd USENIX Workshop on Electronic Commerce"*, 1996.