Computationally sound compositional logic for key exchange protocols *

Anupam Datta, Ante Derek, John C. Mitchell
Dept. Computer Science, Stanford University
{*danupam, aderek, mitchell*}*@cs.stanford.edu*

Bogdan Warinschi
Loria, INRIA-Lorraine
warinsch@loria.fr

Abstract

We develop a compositional method for proving cryptographically sound security properties of key exchange protocols, based on a symbolic logic that is interpreted over conventional runs of a protocol against a probabilistic polynomial-time attacker. Since reasoning about an unbounded number of runs of a protocol involves induction-like arguments about properties preserved by each run, we formulate a specification of secure key exchange *that is closed under general composition with steps that use the key. We present formal proof rules based on this game-based condition, and prove that the proof rules are sound over a computational semantics. The proof system is used to establish security of a standard protocol in the computational model.*

1 Introduction

Key exchange protocols enable secure communication over an untrusted network by setting up shared keys between two or more parties. For example, SSL [28] and TLS [21] provide symmetric encryption keys for secure Internet transactions, IPSec [36] protocols provide confidentiality and integrity at the IP layer, IEEE 802.11i [1] provides data protection and integrity in wireless local area networks, and Kerberos [37] provides authenticated client-server interaction in local area networks. While some of these protocols have been proved correct in the simplified symbolic Dolev-Yao model [33, 38, 44, 27], most key exchange protocols in use today have not been proved secure in the complexity-theoretic model of modern cryptography. Our aim is to develop a formal logic that will allow us to convert known proofs based on the Dolev-Yao model into formal proofs that are provably sound for the standard cryptographic interpretation based on probabilistic polynomial-time attacks. This paper presents progress on key exchange protocols, in the form of an axiom system for relevant primitives, a soundness proof for these rules, and a condition on key exchange that can be proved invariant under steps that legitimately use an agreed key for its intended purpose.

In proving security of a key exchange protocol, it is necessary to state an appropriate security property, one that is true of the protocol, and sufficient to guarantee that the key is suitable for use. Several approaches have been proposed in the cryptographic literature [11, 10, 45, 14], including the concept of *key indistinguishability:* a key produced by a key exchange protocol should be indistinguishable (given access to messages sent in the protocol) from one chosen at random from the same distribution. This is a very natural condition, and certainly a desirable goal for key exchange protocols. However, key indistinguishability does not appear satisfactory for incremental verification of some important protocols, or for stating the property achieved when the key exchange steps are combined with protocols that use the key. The somewhat weaker condition we use instead works well in our formal setting and has the advantage that it may be useful when the key exchange protocol contains key confirmation steps (as in SSL) that may prevent the key exchange protocol from satisfying the stronger key indistinguishability condition.

In this paper, we develop a compositional method for proving cryptographically sound security properties of key exchange protocols, develop a suitable specification of acceptable key generation, and apply the method to an illustrative sample protocol. We use a symbolic logic for specifying security and a formal proof system for proving security properties of key exchange protocols. The specific logic we use in this paper builds on a computational version [20] of *Protocol Composition Logic (PCL)* [23, 24, 18, 19, 33] and offers several advantages. First, the analyst may reason abstractly, without referring to probability, asymptotics, or actions of an attacker. At the same time, a proof provides the same mathematical guarantees as more detailed reduction-based proofs because the semantics of *Computational PCL* is defined with respect to a complexity-theoretic model of protocol execution, and the axioms and proof rules are proved sound using conventional reduction ar-

*This research was supported by NSF grants PORTIA and TRUST, by the ONR SPYCE project, and by the French ARA SSIA Formacrypt and ACI Jeunes Chercheurs JC9005.

guments. This framework is also flexible enough to treat cryptographic primitives like CMA-secure signatures and complexity-theoretic assumptions like Decisional Diffie-Hellman naturally as axioms in the proof system. Second, PCL comes with *composition theorems* [18, 19], which also carry over to Computational PCL. These theorems allow the security proof of a composite protocol to be built up from proofs of its parts, without implicit assumptions like disjoint state of repeated instances of the protocol. This compositional approach is useful for handling large, complex protocols like IEEE 802.11i [33] and is relevant to key exchange protocols since key exchange is intended for use in composition with other protocols. Finally, since the proofs are completely axiomatic, in principle they can be machine-checked, although we currently do not have an implementation that allows this.

We demonstrate the applicability of the proof method by formalizing and proving some security properties of the ISO-9798-3 key exchange protocol [39] and its composition with a canonical secure sessions protocol. The security proof for ISO-9798-3 relies on the Decisional Diffie-Hellman assumption and the use of CMA-secure signatures, while the security of the secure sessions protocol relies on the use of a CPA-secure symmetric encryption scheme [30]. The fact that these two protocols compose securely when executed one after the other follows from the *sequential composition theorem* [19]. In order to model and prove security for these protocols, we had to extend the computational model and logic [20] to include a number of additional cryptographic primitives: signatures, symmetric encryption, as well as codify the Decisional Diffie-Hellman assumption. This application provides evidence that Computational PCL can support axiomatic proofs of interesting security protocols. The results of the present paper open the way to developing computationally sound proofs of larger practical protocols like IEEE 802.11i and Kerberos.

Organization Section 2 presents our security model for key exchange and secure sessions and compares it to other models in the literature. Section 3 describes the programming language for representing protocols and defines their execution model. Section 4 presents the logic for expressing protocol properties and the proof system for proving such properties. Section 5 presents the application of the method to the ISO-9798-3 protocol, a generic secure sessions protocol, and the proof of their secure composition. Section 6 presents the semantics of extensions to the logic. Section 7 compares our model for key exchange to other models in the literature. Finally, Section 8 concludes the paper and discusses directions for future work.

2 Security Model

In this section, we describe some problems with inductive compositional reasoning about key indistinguishability and present an alternative condition that appears more suitable for our purposes. We also give a definition of secure session that uses a key. These definitions are formulated within a complexity-theoretic model, in the style of modern cryptographic game-based definitions [11]. In subsequent sections, we use these definitions to develop a cryptographically sound proof system for reasoning about key exchange protocols. The soundness proofs are subtle and involve complexity-theoretic reductions.

2.1 Key indistinguishability

Our goal is to develop a formal method for compositional proofs of protocol suites involving key exchange protocols. Since we intend to apply our method to prove security properties of existing protocols in common use, as presented in RFCs and as currently implemented, we aim to accommodate conditions such as semantic security that may be weaker than the security conditions advocated by some cryptographers. A central concept in many compositional proof methods [4, 25, 26, 13, 19] is an *invariant*. In developing compositional security proofs of complex protocols [33], we require that each protocol component respects the invariants of the other components in the system [19].

It appears that standard cryptographic security definitions for key exchange like *key indistinguishability* [11, 10] are *not* invariant in the manner needed for an inductive, compositional proof of security of many systems that include both key exchange and use of the key. of Even if a key exchange protocol, run by itself in isolation, produces a key that is indistinguishable from a random value chosen from the same distribution, key indistinguishability is generally lost as soon as the key is used to encrypt a message of a known form or with partially known possible content. Moreover, some situations allow one agent to begin transmitting encrypted data before the other agent finishes the last step of the key exchange, rendering key indistinguishability false at the point that the key exchange protocol finishes. (This appears to be the case for SSL [28]; see [42] for a discussion of data transfer before the key exchange *finished* messages are received.) Furthermore, some key exchange protocols even use the generated key during the protocol, preventing key indistinguishability. Fortunately, many protocols that use keys do not require key indistinguishability to provide meaningful security guarantees. In particular, semantic security [31] does *not* require that the keys used remain indistinguishable from random.

To circumvent the technical problems we encountered in working with key indistinguishability, we develop an alter-

native notion that is parameterized by the security goal of the application in which the resulting key is used. As concrete examples, we consider cases where the key is used for encryption or MAC. The security definition for key exchange requires that the key produced is "good" for that application, i.e. an adversary interacting with the encryption scheme using this key cannot win the security game for that scheme (for example, the IND-CPA game for encryption). The resulting definition for key exchange is invariant under composition with the application protocol which uses the key.

Some interesting issues related to key indistinguishability were raised by Rackoff, who proposed an example protocol that is explained in the appendix of [14]. Rackoff's example illustrates the difference between key indistinguishability based on an attacker who uses a challenge (the key or a surrogate chosen randomly from the same distribution) to interact with subsequent steps of the key exchange protocol, and indistinguishability based on an attacker who cannot execute further protocol steps. The definition of key usability that we use in this paper is similar to the weaker notion of key indistinguishability in that the adversary who attempts to win, for example, the IND-CPA game for encryption, does not have the opportunity to interact further with other protocol participants. On the other hand, because protocols (such as SSL [28]) that provide key confirmation steps will also fail the stronger form of definition suggested by Rackoff's example, we consider the weaker condition we use advantageous for certain practical settings. Specifically, we believe that whatever security properties are enjoyed by practical key exchange protocols in current use, there is merit in being able to state and prove these properties precisely. We also believe that the setting presented in this paper provides a useful starting point for expressing and reasoning about stronger security conditions.

We emphasize that the definition and subsequent theorems below apply to any cryptographic primitive that satisfies the application game condition, e.g., the **ENC** axiom (presented in Section 4.2) holds for any encryption scheme that satisfies the IND-CPA condition.

2.2 Key usability

While there are many desirable properties a "good" key exchange protocol might satisfy, such as key freshness, high key entropy, and agreement, one essential property is that the key should be suitable for use. Specifically, an adversary who interacts with the key exchange protocol should not be able to extract information that can compromise the application protocol which uses the resulting key. This is the main idea underlying the security definition that we develop in this section. The specific applications that we focus on here are symmetric encryption and message authentication codes. But the definition can be extended in a natural manner to cover other primitives. There are several ways in which this intuition can be translated into a security definition. We present one such definition below.

We define usability of keys obtained through a key exchange protocol Σ with respect to a class of applications S via a two-phase experiment. The experiment involves a two-phase adversary $\mathcal{A} = (\mathcal{A}_e, \mathcal{A}_c)$. In the *key exchange phase*, the honest parties run sessions of the protocol following the standard execution model: each principal executes multiple sessions of the protocol (as both initiator and responder) with other principals; the communication between parties is controlled by the adversary $\mathcal{A}_e$. At the end of the key exchange phase, the adversary selects a challenge session sid among all sessions executed by the honest parties, and outputs some state information St representing the information $\mathcal{A}_e$ was able to gather during its execution. Let k be the key locally output by the honest party executing the session sid. At this point, the experiment enters its second phase—the *challenge phase* where the goal of the adversary is to demonstrate an attack against a scheme $\Pi \in S$ which uses the key k. After $\mathcal{A}_c$ receives as input St, it starts interacting with Π according to the game used for defining security of the application protocols in S. For example, if S is a set of encryption schemes, then the relevant game may be IND-CPA, IND-CCA1, or IND-CCA2 [30]. Since the specific task we treat in this paper is secure sessions, we formalize the case when the game defines IND-CPA security. Thus, in playing the game, $\mathcal{A}_c$ has access to a left-right encryption oracle under k, and in addition, it receives as input the state information from $\mathcal{A}_e$. The advantage of the adversary is defined as for the standard IND-CPA game with the difference that the probability is taken over the random coins of the honest parties (used in the execution of the protocol), the coins of the two adversaries, and the coins used for encryption in the challenge phase. The keys obtained by running the key exchange protocol are usable for the schemes in S if this advantage is bounded above by a negligible function of the security parameter, for *all* encryption schemes in S. The universal quantification over schemes is used to capture the fact that the security property is guaranteed for all encryption schemes which satisfy the IND-CPA condition. This idea is formally stated as Definition 1 below.

We note that in this definition the adversary $\mathcal{A}_c$ is not allowed to interact with the key exchange protocol in the challenge phase. In subsequent work, we plan to explore variants of this definition with other forms of communication between the adversaries $\mathcal{A}_e$ and $\mathcal{A}_c$. Note also that this setting does not consider dynamic corruption and gives the adversary access to one key from one session, as opposed to multiple keys from multiple sessions.

Definition 1. *Consider the following experiment*

$\mathbf{Exp}^{b}_{\mathcal{A},\Sigma,\Pi}(\eta)$*, involving an adversary* $\mathcal{A} = (\mathcal{A}_e, \mathcal{A}_c)$*, a key exchange protocol* Σ *and an encryption scheme* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$*. The experiment is parameterized by the bit* b*.*

- *The adversary* $\mathcal{A}_e$ *is given as input the security parameter and can make the following queries:*

  – Request that a new principal i *be added to the system: new pairs of encryption/decryption keys are generated for the principal via* $(pk_i, sk_i) \xleftarrow{\$} \mathcal{K}(\eta)$*. The principal is willing to engage in any number of sessions of the key exchange protocol as both initiator and responder, with any other principal in the system.*

  – Send a message m *to a protocol session: the receiving party processes* m *and returns to* $\mathcal{A}_e$ *the answers it computes according to the protocol.*

  – At some point $\mathcal{A}_e$ *finishes its execution and outputs* (sid, St)*, that is a session identifier and some state information.*

- *Adversary* $\mathcal{A}_c$ *is given the state information* St *and is provided access to a left-right encryption oracle* $\mathcal{E}(LR(\cdot,\cdot,b),k)$ *keyed with the key* k *locally output in session* sid*.*

- *Adversary* $\mathcal{A}_c$ *plays the standard IND-CPA game: it submits pairs of equal-length messages* (m_0, m_1) *to the encryption oracle and obtains in return* $\mathcal{E}(m_b, k)$*.*

- *At some point* $\mathcal{A}_c$ *finishes its execution and outputs a guess bit* d*, which is also the output of the experiment.*

For any adversary $\mathcal{A} = (\mathcal{A}_e, \mathcal{A}_c)$ *we define its advantage as:*

$$\mathbf{Adv}^{ke}_{\mathcal{A},\Sigma,\Pi}(\eta) = Pr[\mathbf{Exp}^{1}_{\mathcal{A},\Sigma,S}(\eta) = 1] - Pr[\mathbf{Exp}^{0}_{\mathcal{A},\Sigma,S}(\eta) = 1]$$

and say that keys exchanged through Σ *are usable in schemes in* S *if for any* $\Pi \in S$ *the advantage of any probabilistic, polynomial-time adversary* $\mathcal{A}$ *is negligible.*

The definition can be easily modified to define a similar usability property of keys for other primitives, for example, message authentication codes, by appropriately changing the security game that is played in the second phase.

The above definition of usability is consistent with accepted definitions of symmetric key-based primitives based on security against adversaries that are allowed arbitrary uses of the primitive in a priori unknown settings. In addition, our model considers the possibility that key generation is accomplished using a key exchange protocol instead of a non-interactive algorithm. The adversary is provided with auxiliary information obtained by interacting with this protocol.

2.3 Logical formalization

Key exchange The game described in section 2.2 is used to define the semantics of a "Goodkey" predicate and to provide the basis for computational proofs of the soundness of formal axioms using this predicate. In fact, the basic predicate of the logic is GoodKeyAgainst, which asserts that a key is good against a specific agent; the truth of this predicate at any point in the run of one or more protocols will depend on the additional information available to that agent from observations about the protocol steps and actions of any protocol adversary. In logical proofs involving key exchange protocols and their use, we use a derived predicate SharedKey, which asserts that a key is good against any agent not among those sharing the key. (The agents who share the key are arguments to the predicate.)

Secure sessions Formulas involving SharedKey are also used to reason about protocols that use a key generated by a key exchange protocol. A key obtained by running a key exchange protocol may be used to encrypt and authenticate subsequent communication in what is commonly referred to as a *secure session* or *secure channel*. As an example, we will use a formula involving SharedKey as a precondition to a proof of security of a simple one-message protocol in which the sender encrypts some data using a symmetric encryption scheme. Such a session provides *secrecy* if, assuming the sender flips a coin and sends one of two known messages m_0 or m_1 depending on the outcome, the attacker's probability of determining which message was sent is close to $1/2$. We express these additional probabilistic properties using other predicates of Computational PCL. While secure sessions typically provide integrity and replay protection guarantees, in this paper, we only focus on the secrecy property.

Composition We obtain a security proof of the composition of key exchange with secure sessions using a general composition theorem [19]. In carrying out the composition, we ensure that the two protocols do not degrade each other's security guarantees. Technically, this is accomplished by checking that each component respects the invariants of the other. It is important to notice the distinction between the "conditional" composition used in this logic and "universal" composition approaches [12, 8]. Specifically, the PCL composition theorems only apply when a protocol is composed with protocol steps that respect specified invariants. No guarantees are expressed or implied for composition with protocols that violate invariants that are needed.

Because of the requirements on composition, some provable properties may only hold under relatively stringent conditions. In our framework, it may be possible to prove security properties of a key exchange protocol, assuming

$$
\begin{aligned}
\mathbf{Init}(\tilde{Y}) \equiv\ & [\\
& \texttt{new}\ x; \\
& gx := \texttt{expg}\ x; \\
& \texttt{send}\ \hat{X}, \hat{Y}, gx; \\
& \texttt{receive}\ \hat{Y}, \hat{X}, z, s; \\
& \texttt{verify}\ s, (z, gx, \hat{X}), \hat{Y}; \\
& r := \texttt{sign}\ (gx, z, \hat{Y}), \hat{X}; \\
& \texttt{send}\ \hat{X}, \hat{Y}, r; \\
& k := \texttt{dhkeyken}\ x, z; \\
]_{\tilde{X}} &
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Resp} \equiv\ & [\\
& \texttt{receive}\ \hat{X}, \hat{Y}, w; \\
& \texttt{new}\ y; \\
& gy := \texttt{expg}\ y; \\
& r := \texttt{sign}\ (gy, w, \hat{X}), \hat{Y}; \\
& \texttt{send}\ \hat{Y}, \hat{X}, gy, r; \\
& \texttt{receive}\ \hat{X}, \hat{Y}, t; \\
& \texttt{verify}\ t, (w, gy, \hat{Y}), \hat{X}; \\
& k := \texttt{dhkeyken}\ y, w; \\
]_{\tilde{Y}} &
\end{aligned}
$$

Figure 1. Roles of the ISO-9798-3 protocol

certain invariants, even if the key exchange protocol is augmented to provide a decryption oracle after one party has completed the protocol. However, the condition given in Definition 1 must be understood as only guaranteeing usefulness of the key material itself, apart from some aspects of interaction between the key exchange protocol and protocols that use the key. In our framework, the way the key is used is handled by the composition theorems. In particular, a key exchange protocol that provides a decryption oracle will generally not satisfy the requirements of the composition theorem since the subsequent key usage protocol will typically require that no such mechanisms are present.

3 Modeling Protocols

We use a simple "protocol programming language" based on [24, 18, 19] to represent a protocol by a set of roles, such as "Initiator", "Responder" or "Server", each specifying a sequence of actions to be executed by a honest participant. Protocol actions include nonce generation, signature creation and verification, pattern matching, and communication steps (sending and receiving). The actions `expg` and `dhkeyken` are used to create a public Diffie-Hellman key ($v := g^x$), and to create a key based on a Diffie-Hellman public/private key pair ($v := KeyGen(PRF(y^x))$), respectively. In Table 1 we show how to specify the roles of the ISO-9798-3 protocol using this language.

As usual, we consider a two-phase execution model. In the initialization phase of protocol execution, we assign a set of roles to each principal, identify a subset which is honest, and provide all entities with keys for digital signature creation/verification, a public group generator for the Diffie-Hellman key exchange, and random coins. In the execution phase, the adversary executes the protocol by interacting with honest principals. We make the standard assumption that the adversary has complete control over the network, i.e. it sends messages to the parties and intercepts their answers, as in the accepted cryptographic model of [11].

Informally, a formula of the logic expresses a property of all *runs* of a protocols, where a run is a record of all actions executed by honest principals and the attacker during protocol execution. Since honest principals execute symbolic programs, a run contains symbolic descriptions of the actions executed by honest parties as well as the mapping of bitstrings to variables. On the other hand, although the attacker may produce and send arbitrary bitstrings, the run only records the send-receive actions of the attacker, and not its internal actions. The execution model is presented in more detail in an earlier paper [20].

4 Protocol Logic

In this section, we present relevant parts of the syntax and proof system of Computational PCL [20]. The syntax indicates the kinds of protocol security properties that are expressible in the logic. The proof system is used for axiomatically proving such properties for specific protocols. It includes axioms capturing properties of cryptographic primitives like signature and encryption schemes, which are used as building blocks in protocol security proofs.

4.1 Syntax

The formulas of the logic are given in Table 1. Protocol proofs usually use modal formulas of the form $\psi[P]_{\tilde{X}}\varphi$. The informal reading of the modal formula is that if $\tilde{X}$ starts from a state in which ψ holds, and executes the program P, then in the resulting state the security property φ is guaranteed to hold irrespective of the actions of an attacker and other honest agents. Many protocol properties are naturally expressible in this form (see Section 5 for examples). Most formulas have the same intuitive meaning as in the symbolic model [18, 19], except for predicates Indist and

Action Predicates:

$$\mathsf{a} ::= \mathsf{Send}(T,t) \mid \mathsf{Receive}(T,t) \mid \mathsf{Verify}(T,t,N) \mid \mathsf{Sign}(T,t) \mid \mathsf{Encrypt}(T,t,k) \mid \mathsf{Decrypt}(T,t,k) \mid \mathsf{New}(T,n)$$

Formulas:

$$\varphi ::= \mathsf{a} \mid t = t \mid \mathsf{Start}(T) \mid \mathsf{Indist}(T,\ t) \mid \mathsf{GoodKeyAgainst}(T,\ t) \mid \mathsf{Fresh}(T,\ t) \mid \mathsf{Honest}(N) \mid \mathsf{Contains}(t,\ t) \mid \mathsf{DHSource}(T,\ t) \mid \mathsf{PSource}(T,\ n,t,t) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists V.\varphi \mid \forall V.\varphi \mid \neg\varphi \mid \varphi \supset \varphi \mid \varphi \Rightarrow \varphi$$

Modal formulas:

$$\Psi ::= \varphi\ [Strand]_T\ \varphi$$

Table 1. Syntax of the logic

GoodKeyAgainst. We describe the meaning of standard formulas informally below, and give a precise semantics in a later section. Predicates that are most relevant to the key exchange example are presented alongside proof rules in the next subsection.

For every protocol action, there is a corresponding action predicate which asserts that the action has occurred in the run. For example, $\mathsf{Send}(\tilde{X},t)$ holds in a run where the thread $\tilde{X}$ has sent the term t. Action predicates are useful for capturing authentication properties of protocols since they can be used to assert which agents sent and received certain messages. $\mathsf{Fresh}(\tilde{X},t)$ means that the value of t generated by $\tilde{X}$ is "fresh" in the sense that no one else has seen any messages containing t, while $\mathsf{Honest}(\hat{X})$ means that $\hat{X}$ is acting honestly, *i.e.*, the actions of every thread of $\hat{X}$ precisely follow some role of the protocol.

4.2 Proof System

The proof system used in this paper is based on the proof system for the symbolic execution model developed in [18, 19, 5]. A first step towards developing a proof system faithful to the complexity-theoretic semantics is given in [20]. In this section, we describe the predicates and axioms for reasoning about Diffie-Hellman, symmetric encryption, and signature primitives introduced in this paper. The soundness theorem for the extended proof system is in Section 6. We reiterate that the advantage of using the proof system is that its justification using cryptographic-style arguments is a one-time mathematical effort; protocol proofs can be carried out symbolically using the proof system without explicitly reasoning about probability and complexity.

Diffie-Hellman key exchange: Reasoning about Diffie-Hellman key exchange can be divided into two steps. First we reason about symbolic actions of honest participants; then we deduce computational key secrecy properties from the fact that honest principals follow certain rules when dealing with Diffie-Hellman key material.

The DHSource predicate is used to reason about the source of a piece of information, such as a nonce. Intuitively, the formula $\mathsf{DHSource}(\tilde{X},x)$ means that the thread $\tilde{X}$ created the nonce x, and in all subsequent actions of that thread it appears only in `expg` and `dhkeyken` actions. In other words, $\mathsf{DHSource}(\tilde{X},x)$ holds if a thread only uses x "inside" exponential g^x or a key $k = KeyGen(PRF(y^x))$.

We extend the proof system with the following axioms used for reasoning about the DHSource axiom. The second is written with a side condition involving the subterm relation $\subseteq$.

S0 $\quad \top\ [\texttt{new}\ x]_{\tilde{X}}\ \mathsf{DHSource}(\tilde{X},x)$

S1 $\quad \mathsf{DHSource}(\tilde{X},x)[\mathsf{a}]_{\tilde{X}}\ \mathsf{DHSource}(\tilde{X},x)$
where ($x \not\subseteq a$ or $\mathsf{a} = \texttt{expg}\ x$
or $\mathsf{a} = \texttt{dhkeyken}\ y, x$ and $x \not\subseteq y$)

Axioms **S0** and **S1** model introduction and persistence of the DHSource predicate. Informally, after a thread $\tilde{X}$ creates a new nonce x, $\mathsf{DHSource}(\tilde{X},x)$ holds (axiom **S0**), and it continues to hold (axiom **S1**) as long as the thread does not use x, other than creating a public key g^x, and creating and using a shared key $v = y^x$. Axioms **S0** and **S1** capture simple properties about information flow within a program and we prove their soundness using direct arguments. When we use these axioms in a formal proof, we are essentially performing induction over symbolic actions of honest parties proving that they treat Diffie-Hellman exponents in a correct way.

The result of a good key exchange protocol is a shared key k which is indistinguishable from a randomly chosen key by a polynomial-time attacker. As discussed in the introduction, after the key is used (e.g. to establish a secure session), partial information about the key is revealed to the attacker. In our model we capture the quality of a key using the predicate GoodKeyAgainst. Informally we wish to capture the property that the key can be securely used for encryption, even if the attacker has some partial information about the key. A bit more formally, $\mathsf{GoodKeyAgainst}(\tilde{X},k)$ holds whenever no probabilistic polynomial-time algorithm, given $\tilde{X}$'s view of the run, can win the IND-CPA game if the challenger uses key k instead of a key generated using the key generation algorithm. We often use the shorthand $\mathsf{SharedKey}(\tilde{A},\tilde{B},k)$ to denote, that k is a good key against everyone except $\tilde{A}$ and $\tilde{B}$. More precisely $\mathsf{SharedKey}(\tilde{A},\tilde{B},k) \equiv \forall\tilde{X}(\tilde{X} = \tilde{A} \vee \tilde{X} = \tilde{B} \vee \mathsf{GoodKeyAgainst}(\tilde{X},k))$.

We extend the proof system with the following axiom

used for establishing key secrecy in Diffie-Hellman key exchange.

DH $\mathsf{Honest}(\hat{X},\hat{Y}) \wedge \mathsf{DHSource}(\tilde{X},x) \wedge \mathsf{DHSource}(\tilde{Y},y)$
$\wedge \mathsf{KeyGen}(\tilde{X},k,x,g^y) \Rightarrow \mathsf{SharedKey}(\tilde{X},\tilde{Y},k)$

Axiom **DH** says that if two threads of honest agents $\tilde{X}$ and $\tilde{Y}$ use their respective private key in a safe way, then the shared key generated from g^{xy} can be safely used with any IND-CPA secure encryption scheme. Note that this axiom does not capture key agreement: threads $\tilde{X}$ and $\tilde{Y}$ could in principle have different keys (or no keys at all). Key agreement has to be established by other methods. We prove the soundness of axiom **DH** using a cryptographic-style reduction to the security of the underlying IND-CPA secure encryption scheme and to the Decisional Diffie-Hellman assumption. The proof employs a hybrid argument and is sketched in Section 6.

Signatures: The signature axiom is given below.

SIG $\mathsf{Verify}(\tilde{X},m,\hat{Y}) \wedge \mathsf{Honest}(\hat{X},\hat{Y}) \Rightarrow \exists \tilde{Y}.\mathsf{Sign}(\tilde{Y},m)$

Informally, this axiom says that if a thread $\hat{X}$ performs a successful signature verification step using a public key of an honest party $\hat{Y}$ then there has to be a thread $\tilde{Y}$ of agent $\hat{Y}$ that performed the signature operation on the same message. This axiom captures unforgeability of signatures and its soundness is proved by reduction to the CMA-security game for signatures. A complete proof will appear in the full version of this paper.

Symmetric Encryption: In the symbolic model [18, 19], the predicate Has states that a principal can "derive" a message or its contents from the information gathered during protocol execution. Since secrecy in the computational model involves absence of any partial information, we use the predicate $\mathsf{Indist}(\tilde{X},t)$ to state that no probabilistic polynomial-time algorithm, given $\tilde{X}$'s view of the run, can distinguish the actual bitstring corresponding to the term t from a random bitstring chosen from the same distribution.

The $\mathsf{PSource}(\tilde{X},b,m,k)$ predicate means that the bit b and the message m were chosen by $\tilde{X}$ via a `pick` action, and in all subsequent actions of that thread b does not appear, and m appears only inside encryptions with the key k. We use it for expressing security properties of symmetric encryption schemes and reasoning about protocols which use such schemes.

We extend the proof system with the following axioms used for reasoning about symmetric encryption.

PS0 $\top\ [(m,b) = \mathtt{pick}\ m_0,m_1]_{\tilde{X}}\ \mathsf{PSource}(\tilde{X},b,m,k)$

PS1 $\mathsf{PSource}(\tilde{X},b,m,k)[\mathsf{a}]_{\tilde{X}}\ \mathsf{PSource}(\tilde{X},b,m,k)$
$(m,b \not\sqsubseteq \mathsf{a}$ or $\mathsf{a} = \mathtt{enc}\ m,k)$

ENC $\mathsf{PSource}(\tilde{X},b,m,k) \wedge \mathsf{Honest}(\hat{X},\hat{Y}) \wedge$
$\mathsf{SharedKey}(\tilde{X},\tilde{Y},k) \wedge$
$(\mathsf{Decrypts}(\tilde{Y},m,k) \wedge \mathsf{Contains}(m,m') \wedge$
$\mathsf{Send}(\tilde{Y},m'') \supset \neg\mathsf{Contains}(m'',m')) \wedge$
$(\mathsf{Decrypts}(\tilde{X},m,k) \wedge \mathsf{Contains}(m,m') \wedge$
$\mathsf{Send}(\tilde{X},m'') \supset \neg\mathsf{Contains}(m'',m')) \wedge$
$\wedge \tilde{Z} \neq \tilde{X} \wedge \tilde{Z} \neq \tilde{Y} \Rightarrow \mathsf{Indist}(\tilde{Z},b)$

Axiom **ENC** captures the properties of an IND-CPA encryption scheme. Informally, in a scenario where k is a shared key between threads $\tilde{X}$ and $\tilde{Y}$, if $\tilde{X}$ chooses a message m and the bit b via a `pick` action, and both threads follow the rules of the IND-CPA game (i.e. the do not send parts of messages they decrypt) then the bit b should be indistinguishable from a random bit to any other party.

Discussion: The presented axioms deduce computational properties based on symbolic actions executed by individual honest parties. This resembles the setup in defining security properties of cryptographic primitives using games. For example, in the IND-CPA game the challenger is required to generate a random key and use it for encryption only. If this syntactic property is satisfied, then the security condition (semantic security) is guaranteed to hold for all computational adversaries interacting with the challenger. The predicates DHSource and PSource are used to exactly state symbolic constraints for actions of honest parties. The axioms **DH** and **ENC** bridge the gap between the symbolic and computational world and are proven sound by reduction to security properties of corresponding primitives.

In Section 2.1 we pointed out that the key indistinguishability property is not invariant under composition. Specifically, focusing on the Diffie-Hellman example, we could have formulated the **DH** axiom to guarantee key indistinguishability by modifying the DHSource predicate to preclude the case where the resulting secret is used as a key. The resulting axiom could be proven sound by a similar reduction. However, this axiom will not be useful in a proof involving a composition of a key exchange protocol with a protocol that uses a key.

5 Applications

5.1 Key Exchange

In this section, we use the protocol logic to formally prove a property of the ISO 9798-3 protocol. The ISO

9793-3 protocol is defined by a set of two roles $Q_{ISO} = \{\mathbf{Init}, \mathbf{Resp}\}$, comprising one role $\mathbf{Init}$ for the initiator of the protocol and one program $\mathbf{Resp}$ for the responder. The roles of the protocol, written using the protocol language are given in Figure 1.

Writing $\mathbf{Init} = \mathbf{Init}(\tilde{Y})$ for the protocol steps of initiator $\hat{X}$ communicating with responder $\hat{Y}$, the security guarantee for the initiator is expressed by the following formula:

$$Q_{ISO} \vdash \top\, [\mathbf{Init}]_{\tilde{X}}\, \mathsf{Honest}(\hat{X}, \hat{Y}) \supset \exists \tilde{Y}.\exists y.\ z = g^y \wedge \mathsf{SharedKey}(\tilde{X}, \tilde{Y}, k)$$

This formula states the key usability property for the initiator when ISO-9798-3 protocol is run in isolation. More precisely, after the initiator $\hat{X}$ completes the initiator steps with $\hat{Y}$ then, assuming both agents are honest in all of their threads, there is one thread $\tilde{Y}$ of agent $\hat{Y}$ that has established a shared key with $\hat{X}$. The meaning of the predicate SharedKey in this formula is defined using the game condition explained in Section 2. Note that this property is guaranteed against any polynomial time adversary when any number of sessions are executed concurrently. The only additional assumption is that honest agents follow roles of the ISO-9798-3 protocol.

The formal proof, given in Table 2, illustrates some general properties of our method. This example proof is modular, consisting of three distinct parts. In the first part of the proof, steps (1)-(2), we establish that value x generated by the initiator is only used to create g^x and the final key g^{xy}. The reasoning in this step is non-cryptographic, and only relies on the structure of the program of the initiator. In the second step the analogous property for y is established based on the security of the signature scheme, and the structure of the responding program. Finally, the axiom that captures the DDH assumption is used to conclude that the key derived from g^{xy} is secure. Notice that the axioms **SIG** and **DH** are used independently to establish different security properties. The two properties are only combined in the last step of the proof.

The modular symbolic proof can be compared with conventional computational arguments, such as the computational proof of the same property by reduction. The reduction proof starts from an adversary against the key derived from g^{xy} and constructs two different adversaries, one against the DDH assumption and the other one against the signature scheme. The assumptions on signatures and DDH are used intertwined. Specifically, to argue that the adversary against DDH is successful, it is necessary to argue that the values that occur in a possible simulation are created by the honest party, and consequently, to argue that the signature scheme is secure. More generally, the analysis of protocols that use more primitives, and where the reduction uses multiple adversaries, the proofs become increasingly complex. In contrast, evidence drawn from work with the symbolic version of the logic indicates that axiomatic proofs are at the same level of complexity as our proof for the ISO-9798-3 protocol [1].

5.2 Secure Sessions

We formalize the definition of secure sessions presented in Section 2.3 for a protocol $Q_{SS} = \{\mathbf{InitS}, \mathbf{RespS}\}$ below.

$$Q_{SS} \vdash \Big[(m,b) = \mathtt{pick}\, m_0, m_1; \mathbf{InitS}(\tilde{\mathbf{Y}}, \mathbf{m})\Big]_{\tilde{X}} \mathsf{Honest}(\hat{X}, \hat{Y}) \wedge \tilde{Z} \neq \tilde{X} \wedge \tilde{Z} \neq \tilde{Y} \Rightarrow \mathsf{Indist}(\tilde{Z}, b)$$

In words, this formula states that if initiator $\hat{X}$ picks one of two messages at random and executes the secure sessions protocol with $\hat{Y}$, then the attacker cannot distinguish, which of the two messages was transmitted.

As a concrete example, we consider the secure session protocol with the following initiator program. The responder simply decrypts the message.

$$\mathbf{InitS}(\hat{Y}, m, k) \equiv \Big[e := \mathtt{enc}\ m, k; \mathtt{send}\ \hat{Y}, e;\Big]_{\tilde{X}}$$

Using the proof system we can prove that this protocol provides the secure-session property between threads $\tilde{X}$ and $\tilde{Y}$ assuming that the key k is a shared key between $\tilde{X}$ and $\tilde{Y}$, formally:

$$\mathsf{SharedKey}(\tilde{X}, \tilde{Y}, k) \Big[(m,b) = \mathtt{pick}\, m_0, m_1; \mathbf{InitS}(\tilde{\mathbf{Y}}, \mathbf{m}, \mathbf{k})\Big]_{\tilde{X}} \mathsf{Honest}(\hat{X}, \hat{Y}) \wedge \tilde{Z} \neq \tilde{X} \wedge \tilde{Z} \neq \tilde{Y} \Rightarrow \mathsf{Indist}(\tilde{Z}, b)$$

The security property of an IND-CPA secure encryption scheme (expressed by axiom **ENC**) is central to this proof. This is a point of difference between the logic and the approaches which relate the symbolic and computational models [41, 8] and require stronger cryptographic assumptions such as IND-CCA-2.

5.3 Composition

We prove that the sequential composition of ISO-9798-3 and the secure sessions protocol described above is also a good secure session protocol, when the key generated in the first part is used in the second part. We use the general sequential theorem of [19]. This involves two steps: a) the property guaranteed by ISO (that k is a shared key between $\tilde{X}$ and $\tilde{Y}$) is precisely the assumption of the secure sessions protocol; b) two protocols satisfy each other's invariants (e.g. line (6) of Table 2). This step guarantees that one protocol does not provide an oracle that can be used to break the security of the other protocol (see [19] for further

S0	$\top\ [\texttt{new}\ x;]_{\tilde{X}}\ \mathsf{DHSource}(\tilde{X}, x)$	(1)
$\mathbf{S1}, (1)$	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \mathsf{DHSource}(\tilde{X}, x)$	(2)
AA1	$\top\ [\texttt{verify}\ s, (z, gx, \hat{X}), \hat{Y};]_{\tilde{X}}\ \mathsf{Verify}(\tilde{X}, (z, gx, \hat{X}), \hat{Y})$	(3)
$\mathbf{P}, \mathbf{SEQ}, (3)$	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \mathsf{Verify}(\tilde{X}, (z, gx, \hat{X}), \hat{Y})$	(4)
$\mathbf{SIG}, (4)$	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \exists \tilde{Y}.\mathsf{Sign}(\tilde{Y}, (z, gx, \hat{X}))$	(5)
HON	$\mathsf{Honest}(\hat{Y}) \wedge \mathsf{Sign}(\tilde{Y}, (z, gx, \hat{X})) \supset \exists y.\ z = g^y \wedge \mathsf{DHSource}(\tilde{Y}, y)$	(6)
(6)	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \mathsf{Honest}(\hat{X}, \hat{Y}) \supset \exists \tilde{Y}.\exists y.\ z = g^y \wedge \mathsf{DHSource}(\tilde{Y}, y)$	(7)
$(7), (2)$	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \mathsf{Honest}(\hat{X}, \hat{Y}) \supset \exists \tilde{Y}.\exists y.\ z = g^y \wedge \mathsf{DHSource}(\tilde{X}, x) \wedge \mathsf{DHSource}(\tilde{Y}, y)$	(8)
$(8), \mathbf{DH}$	$\top\ [\mathbf{Init}]_{\tilde{X}}\ \mathsf{Honest}(\hat{X}, \hat{Y}) \supset \exists \tilde{Y}.\exists y.\ z = g^y \wedge \mathsf{SharedKey}(\tilde{X}, \tilde{Y}, k)$	(9)

Table 2. Secrecy proof for the initiator in the ISO-9798-3 Protocol

elaboration and examples of composition). The composition would not have worked if we used key indistinguishability instead of the weaker shared-key property.

Consider a protocol in which both parties execute session of the Q_{ISO} protocol followed by a session of protocol Q_{SS} protocol. Moreover, key used in the secure session protocol is exactly the key established in the key-exchange phase, while the message transmitted in the secure session phase in an arbitrary message obtained via input interface of a resulting protocol. Formally, $Q = \{\mathbf{InitQ}, \mathbf{RespQ}\}$ where the initiators program is given by (responders program is analogous):

$$\mathbf{InitQ}(\hat{Y}, m) \quad \equiv \quad [\mathbf{Init}(\hat{Y}); \mathbf{InitS}(\hat{Y}, k, m)]_{\tilde{X}}$$

To formally prove that the combined protocol also provides secret session property we use composition theorems and the sequencing rules in steps as follows: First we need to ensure that protocols satisfy each others invariants, i.e. $Q_{SS} \vdash \Gamma_{ISO}$ and $Q_{ISO} \vdash \Gamma_{SS}$, where Γ_{ISO} and Γ_{SS} are invariants used in the proof of the key exchange and the secure session protocol respectively. Informally, Γ_{ISO} requires that honest agents treat their Diffie-Hellman exponents correctly, while Γ_{SS} requires that no party possessing the shared key acts as a decryption oracle.

$$\begin{aligned}
\Gamma_{ISO} \quad = \quad & \mathsf{Honest}(\hat{Y}) \wedge \mathsf{Sign}(\tilde{Y}, (z, gx, \hat{X})) \supset \\
& \exists y.\ z = g^y \wedge \mathsf{DHSource}(\tilde{Y}, y) \\
\Gamma_{SS} \quad = \quad & \mathsf{Honest}(\hat{X}, \hat{Y}) \wedge \mathsf{SharedKey}(\tilde{X}, \tilde{Y}, k) \wedge \\
& (\mathsf{Decrypts}(\tilde{Y}, m, k) \wedge \mathsf{Contains}(m, m') \wedge \\
& \mathsf{Send}(\tilde{Y}, m'') \supset \neg \mathsf{Contains}(m'', m')) \\
& (\mathsf{Decrypts}(\tilde{X}, m, k) \wedge \mathsf{Contains}(m, m') \wedge \\
& \mathsf{Send}(\tilde{X}, m'') \supset \neg \mathsf{Contains}(m'', m'))
\end{aligned}$$

When this is established, both proofs are still valid, using the sequencing rule and the fact that the Q_{SS} protocol does not invalidate DHSource predicate we establish that the shared-key property persists from the end of the Q_{ISO} protocol to the end of the Q_{SS} protocol. Finally using the properties of the Q_{SS} protocol, we establish that the combined protocol provides secure session property.

As mentioned in Section 2, there are key exchange protocols which satisfy the key usability property but may not compose well with any subsequent secure session. Using the same example, if a key exchange protocol provides a decryption oracle then Γ_{SS} will not be satisfied and the protocol composition will fail.

6 Computational Semantics and Soundness Theorem

In this section, we outline the main ideas behind the computational semantics and present semantics for the predicates introduced in this paper. We also state the soundness theorem for the proof system and sketch the proof for one representative axiom, demonstrating the connection between validity of logical formulas and standard security definitions of cryptographic primitives. Because of space limitations, the portions of the semantics that are presented in full in [20] are only summarized here, allowing us to describe the semantics of predicates introduced in this paper in more detail. The reader wishing a complete explanation may consult Sections 6 and 7 of [20].

The meaning of a formula is defined with respect to a set of computational traces, where each trace corresponds to one particular execution of the protocol with all the parameters fixed (including the randomness of the attacker and honest parties). Intuitively, the meaning of a formula φ on a set T of computational traces is a subset $T' \subseteq T$ that respects φ in some specific way. For example, an action predicate such as Send selects a set of traces in which a send occurs. The semantics of predicates Indist and GoodKeyAgainst are more complex and involve a second

phase of execution where the distinguisher tries to guess the secret value or break the encryption scheme.

We inductively define the set $[\![\varphi]\!](T, D, \epsilon)$ of traces that is the meaning of formula φ on the set T of traces, with distinguisher D and tolerance ϵ. The distinguisher and tolerance are not used in any of the clauses except for Indist and GoodKeyAgainst, where they are used to determine whether the distinguisher has more than a negligible chance of distinguishing the given value from a random value or wining an IND-CPA game, respectively. A protocol Q will satisfy a formula φ, written $Q \models \varphi$ if for all adversaries, distinguishers, and sufficiently large security parameter, $[\![\varphi]\!](T, D, \epsilon)$ is an overwhelming subset of the set of all possible traces produced by the interaction of protocol Q and attacker A.

Every trace $t \in T$ includes a set of symbolic actions executed by the honest participants, as well as the mapping λ assigning bitstrings to all terms appearing in symbolic actions. A trace t also includes a mapping σ assigning bitstrings to free formula variables. Informally, σ represents the environment in which the formula is evaluated. Technically, as the binding operators (such as quantifiers) are parsed, σ is used to keep track of the values assigned to the variables by these operators.

- $\left|\!\left[\mathsf{DHSource}(\tilde{X}, x) \right]\!\right| (T, D, \epsilon)$ is the collection of all traces $t \in T$ such that for all basic terms y with $\sigma(x) = \lambda(y)$ there is a single symbolic action $(\mathtt{new}\ (\tilde{X}), y)$, and term y does not appear in any symbolic actions except possibly in $(v := \mathtt{expg}\ (\tilde{X}, y))$ and $(v := \mathtt{dhkeyken}\ (\tilde{X}, z, y))$ for some term z different from y.

  Notice that the condition $\sigma(x) = \lambda(y)$ is used to tie the formula variable x to a trace variable y by requiring that they both evaluate to the same bitstring. Therefore, if we fix a particular trace $t \in T$ with an environment σ, then $t \in \left|\!\left[\mathsf{DHSource}(\tilde{X}, x) \right]\!\right| (T, D, \epsilon)$ if in the trace t, the thread $\tilde{X}$ created a new nonce (represented by a trace variable y) with a bitstring value equal to that of $\sigma(x)$, and used the variable y only inside an exponentiation action or a key generation action.

- $\left|\!\left[\mathsf{PSource}(\tilde{X}, b, m, k) \right]\!\right| (T, D, \epsilon)$ is the collection of all traces $t \in T$ such that for all basic terms m', b' with $\sigma(m) = \lambda(m')$ and $\sigma(b) = \lambda(b')$, such that there is symbolic action $((m', b') := \mathtt{pick}\ \tilde{X}, m_1, m_2)$, terms b' and m' do not appear in any symbolic actions except maybe in $(v := \mathtt{enc}\ \tilde{X}, m', k')$, with $\sigma(k) = \lambda(k')$.

- $\left|\!\left[\mathsf{GoodKeyAgainst}(\tilde{X}, k) \right]\!\right| (T, D, \epsilon)$ is the complete set of traces T if the distinguisher D, who is given a complete $\tilde{X}$'s view of the run, has an advantage less than ϵ in winning the IND-CPA game against a challenger using the bitstring corresponding to term k, and empty set $\emptyset$ otherwise. Here the probability is taken by choosing an uniformly random trace $t \in T$ (which includes the randomness of all parties, the attacker as well as the distinguisher randomness).

- $\left|\!\left[\mathsf{Sign}(\tilde{X}, m) \right]\!\right| (T, D, \epsilon)$ is a collection of all traces where $\tilde{X}$ performs a symbolic signing operation on a variable whose bitstring value corresponds to the bitstring value of m.

- $\left|\!\left[\mathsf{Verify}(\hat{X}, m, \hat{Y}) \right]\!\right| (T, D, \epsilon)$ is a collection of all traces where $\tilde{X}$ performs a successful symbolic signature verification operation where the bitstring value of the signed text corresponds to the bitstring value of m, and the bitstring value of the agent name corresponds to the bitstring value of $\hat{Y}$.

Note that the predicates defined by reference to a set of symbolic actions by a thread $\tilde{X}$ only make sense if the agent $\hat{X}$ is honest and therefore its threads only perform symbolic actions. For the threads of dishonest agents, we can define the semantics of these terms arbitrarily. In all provable formulas, these predicates will only occur in conjunction with the assumption that the corresponding agent is honest.

The proof system used in this paper consists of axioms and proof rules presented in [20] extended with the axioms introduced in Section 4.2: **S0**, **S1** and **SH** modeling properties of the Diffie-Hellman key exchange, **SIG** modeling signatures and **PS0**, **PS1** and **ENC** modeling symmetric encryption.

Theorem 1 (Soundness)**.** *The proof system [20] extended with axioms from Section 4.2 is sound for the semantics [20] extended with clauses above.*

This soundness theorem is proved by showing that every axiom is a valid formula and that all proof rules preserve validity. For some axioms and proof rules soundness will follow directly from the execution model or by information theoretic reasoning. Axioms stating properties of cryptographic primitives are proved sound by transforming a protocol and an attacker breaking the axiom to an attacker on the game defining the security property of the cryptographic primitive. Below we give a proof sketch for the soundness of the **DH** axiom (introduced and discussed informally in Section 4.2).

Proof sketch for axiom **DH**. Assuming the axiom is false we deduce that there exists an adversary A, and a distinguisher D such that:

$$|[\![\neg\varphi]\!](T, D, \nu(\eta))| \geq \nu(\eta)$$

is non-negligible (as a function of η). By unwinding the semantics of $\neg\varphi$, we obtain that there exists three parties b_X, b_Y and b_Z such that the set:

$$[\![\neg\varphi]\!] (T, D, \nu(\eta))[\tilde{X} \to b_X][\tilde{Y} \to b_Y][\tilde{Z} \to b_Z]$$

is of non-negligible size in rapport with $|T|$. More precisely, with non-negligible probability, the distinguisher D can successfully break the IND-CPA game played against a standard left-right encryption oracle keyed with the key that party b_X outputs at the end of the protocol execution with party b_Y, provided that D has access to the view associated to party b_Z. Given adversary A and distinguisher D we explain how to construct two adversaries A_1 and A_2, one against the DDH assumption and the other one against the IND-CPA security of the encryption scheme, such that at least one of these two adversaries has non-negligible probability of wining the corresponding security game.

We consider two execution scenarios that involve A and D. In the first scenario adversary A interacts with the protocol as sketched in Section 3. Then, D plays the IND-CPA game in which the key for encryption is the key obtained by party b_X. At the end of the execution D outputs a guess bit d. Let $\mathsf{guess}_1(A, D)$ be the event that the output of D coincides with the bit b of the left-right oracle to which it has access. By the assumption that D is successful we conclude that $P_1 = \Pr[\mathsf{guess}(A, D)_1]$ is non-negligible. (Here, to simplify notation we omit to explicitly show the dependence of the event on the security parameter.)

In the second scenario, the execution of A proceeds also as sketched in Section 3. However, D plays the IND-CPA games against encryption oracles in which the key has been randomly generated (in particular, this key is independent from the execution of the protocol). Let $\mathsf{guess}_2(A, D)$ be the event that D correctly guesses the bit that parameterizes the left-right oracle, and let $P_2 = \Pr[\mathsf{guess}_2(A, D) = b]$.

Intuitively, if the DDH assumption holds, adversary D should not observe any difference between the two different execution scenarios that we consider. Formally, we construct the following adversary A_1 against the DDH assumption. The adversary takes as input a triple $(X = g^x, Y = g^y, Z = g^z)$ and works as follows. It executes adversary A as a subroutine, and emulates for A the behavior of the honest parties. The difference is that for parties b_X and b_Y, the adversary does not generate the values x and y needed for the execution, but whenever it needs to send g^x and g^y it sends X and Y respectively. Notice that here we crucially use that $\mathsf{DHSource}(\hat{X}, x)$ and $\mathsf{DHSource}(\hat{Y}, y)$ hold, since this implies that parties b_X and b_Y only send the values x and y as exponents. (Otherwise, it would be impossible to carry out the simulation).

When A finishes its execution, adversary A_1 flips a bit b and simulates for D the left-right encryption oracle parameterized by b and keyed by the key generated from Z. When D finishes and outputs a bit d adversary A outputs 1 if $d = b$ and 0 otherwise.

Notice that when (X, Y, Z) are such that $z = xy$, the view of (A, D) is as in the normal execution of the protocol, and thus we have that $\Pr[A_1 = 1 | Z = g^{xy}] = \Pr[\mathsf{guess}_1(A, D)]$. When Z is such that z is randomly chosen, the view of (A, D) is as in the alternative execution scenario that we consider, and thus we obtain that $\Pr[A_1 = 1 | Z = g^z] = \Pr[\mathsf{guess}_2(A, D)]$.

The advantage that A_1 has in breaking the DDH assumption is thus:

$$\mathbf{Adv}_{\mathrm{DDH}, A_1}(\eta) = \Pr[\mathsf{guess}_1(A, D)] - \Pr[\mathsf{guess}_2(A, D)] \tag{10}$$

Next, we bound the probability of $\mathsf{guess}_2(A, D)$. Intuitively, if the encryption scheme is IND-CPA secure, no adversary should be able to win the IND-CPA game in the second execution scenario (since the key used in the oracle is a randomly generated key, thus independent from that generated in the key exchange phase). Formally, we construct adversary A_2 against the IND-CPA security of the encryption scheme. The adversary has access to a left-right encryption oracle parameterized by a bit b and proceeds as follows. It runs adversary A as a subroutine and simulates for A the execution of the honest parties involved in the protocol. Thus, it generates the encryption and decryption keys of the honest parties and then receives and outputs messages as prescribed by the protocol. Whenever A finishes its execution, adversary A_2 provides to D the view of party b_Z, whatever state information A has output, and offers access to his own oracle (parameterized by a bit b to be guessed). Notice that if at any point, in order to carry out the simulation adversary A needs to output the encryption of some message m under the key k of the oracle (this is the case for example when the parties exchange confirmation messages using the exchanged key), A_2 can simply submit (m, m) to its encryption oracle.

The guess of A is whatever D outputs. The key observation is that the view of the pair (A, D) is exactly as in the second execution scenario that we consider. Thus A_2 successfully guesses the bit b precisely when, following the execution we have just described, the distinguisher D outputs b. Thus, we obtain that:

$$\mathbf{Adv}_{\text{IND-CPA}, A_2}(\eta) = \\ \Pr[A_2 \text{ wins the IND-CPA game}] = \Pr[\mathsf{guess}_2(A, D)] \tag{11}$$

By Equations (10) and (11) we get that:

$$\Pr[\mathsf{guess}_1(A, D)] = \mathbf{Adv}_{\mathrm{DDH}, A_1}(\eta) + \mathbf{Adv}_{\text{IND-CPA}, A_2}(\eta)$$

Since the left-hand side term is a non-negligible function so is at least one of the summands on the right-hand side. Thus, either the DDH assumption is not valid, or the encryption scheme is not IND-CPA secure. □

7 Related Work

Computational Soundness We use the phrase "computational soundness" to refer to a line of work whose goal is to develop symbolic methods for security analysis faithful to the complexity theoretic model of cryptography. Abadi and Rogaway [2] have initiated this line of research. Their main result is a soundness theorem for a logic of encrypted expressions: a symbolic notion of equivalence between such expressions based on Dolev-Yao deducibility [22] implies computational indistinguishability. This work has been extended to the case of (symbolic) static equivalence [9], while other works investigate completeness aspects of the Abadi-Rogaway logic [40, 29, 3]. All these results hold for a passive adversary, while our results are set in the more general and more realistic framework of active adversaries.

The active setting has been investigated by Backes, Pfitzmann, and Waidner [8] and by Micciancio and Warinschi [41], with further refinements and applications provided in [17, 35, 7]. In these approaches, the core results are emulation theorems that state that the behavior of arbitrary computational adversaries can be emulated by symbolic adversaries. It follows from an emulation theorem that security in the symbolic model implies security in the computational model. However, current emulation theorems require strong cryptographic assumptions (e.g., IND-CCA2 encryption) while the present paper allows weaker assumptions. Our approach appears to offer a higher degree of flexibility and modularity when compared to [8, 41], which requires a new emulation theorem for each added primitive; this may be difficult or impossible in some cases [6]. Similarly, new primitives can be added to the present framework by adding appropriate axioms and proof rules to the logic and proving them sound. However, this appears easier, primarily because it is not necessary to completely axiomatize new primitives, but only to formalize the properties that are needed to prove protocols of interest correct. For instance, our axiom for exponentiation does not explicitly give any algebraic properties (although the soundness proof certainly accounts for them), and only reflects the Decisional Diffie-Hellman assumption.

A complementary line of research is proposed by Impagliazzo and Kapron [34], who provide a logic for reasoning about indistinguishability. Their logic is appropriate for reasoning about security of primitives, but has not been extended to deal with protocols.

An approach similar to the present paper is taken by Gupta and Shmatikov [32] who extend the logic of [20] with signatures and Diffie-Hellman keys, and then use the resulting logic to express security properties of key exchange protocols. The main result is a proof that protocols that satisfy their security requirement are secure with respect to a computational model for secure key exchange due to Shoup [45]. Their logical characterization of secure keys is based on indistinguishability, and unlike our notion is not composable.

Other models of key exchange In previous work, three different approaches have been used to define security of key-exchange protocols: the indistinguishability-based approach [11], the simulation-based security paradigm [45], and universal composability [12] or reactive simulateability [43].

The indistinguishability-based approach was proposed by Bellare and Rogaway [11]. A central aspect of this definition is the notion of *key indistinguishability*, which states that an attacker cannot distinguish between the real key and one chosen at random. This model was refined and extended by Bellare, Petrank, Rackoff and Rogaway (in unpublished work) and later by Canetti and Krawczyk [14]. The approach of Canetti and Krawczyk also offers a limited form of composition guarantees. Specifically, they prove that a key exchange protocol which satisfies their definition can be securely composed with a specific secure sessions protocol, which uses the exchanged key. However, as noted in the introduction of this paper, key indistinguishability is not generally preserved once the key is used. While [14] provides for a *specific* composition, their theorem would not apply, for example, to IEEE 802.11i, where the key exchange protocol (TLS [21]) is composed with a protocol that uses the exchanged key to set up other fresh keys for securing data transmission.

Bellare, Canetti, Krawzyck [10] and Shoup [45] provide simulation-based alternatives. This line of research is grounded in foundational work on secure multi-party computation. Here, security of a real protocol is asserted by comparing it with an ideal protocol, which is secure by construction. As usual with this approach, while the resulting definitions are quite appealing to intuition, security proofs may be quite involved. Moreover, the basic framework of secure multi-party computation does not have built-in compositionality guarantees, and neither of these two models offers improvements with respect to this important aspect.

Finally, the universal composability framework of Canetti [12] has the explicit goal of providing a framework where demonstrated security is preserved under arbitrary composition. Working in this setting Canetti and Krawczyk [15] prove an equivalence between single-session UC-security of key exchange protocols and the indistinguishability-based notion introduced in [14], and their result implies that indistinguishability-based notion may be composable. However, the general compositionality properties offered by the basic UC framework only apply when primitives do not share state. A partial solution is offered in [16], which allows multiple sessions of a protocol to share state under some very specific conditions.

8 Conclusions

We propose and formalize an approach for proving the security of key exchange protocols. The condition we use to express suitability of the key requires that the key should be adequate for the application in which it is used. For example, if a key is used as an encryption key, we ask that an attacker interacting with the protocol cannot use the auxiliary information thus obtained to break an IND-CPA (or IND-CCA) game with that key used in encryption (and decryption) oracles. One important feature of this definition is that the key usability property is an invariant of protocols that use the key, and is therefore amenable to the inductive proofs and the form of composition that we use.

Our security definitions are formalized in a symbolic protocol logic. We extend an existing computational logic [20] with axioms capturing properties of signatures, symmetric encryption, and message authentication codes, as well as with the Decisional Diffie-Hellman assumption. Protocol proofs in this logic are compositional—proofs of compound protocols can be constructed from proofs of their parts. As a simple illustrative example, we show how to compose the proof of ISO-9798-3 with the proof of a secure sessions protocol. Our definition of key usability was crucial for this compositional proof; it could not have been carried out with the key indistinguishability-based definition. The axioms used in a proof identify specific properties of cryptographic primitives that are sufficient to guarantee the desired protocol properties. Specifically, we note that for the secure sessions protocol presented in this paper, an IND-CPA secure encryption scheme is sufficient. This is an important point of difference between our approach and the emulation theorems of [41, 8], since those theorems work only under stronger cryptographic assumptions (e.g., IND-CCA2 for encryption).

Since commonly used reasoning principles are codified in the proof system, protocol security proofs can be carried out at a high-level of abstraction without worrying about probability and complexity. All such details are buried in the proof of the soundness theorem, which is a one-time mathematical effort. The soundness proofs for the various axioms involve standard cryptographic proof techniques. For example, the soundness proof of the signature axiom, **SIG** involves a reduction to the security of CMA-signatures. Among the axioms introduced in this paper, the soundness of the **DH** axiom was the most difficult to establish since it relied on two cryptographic security conditions: Decisional Diffie-Hellman and IND-CPA secure encryption.

We believe that the methods developed in this paper provide a viable alternative to existing methods for carrying out cryptographically-sound security proofs of practical key exchange protocols. In the future, we believe that these methods could be used to prove security properties of protocols like IKEv2 [36], IEEE 802.11i [1], and Kerberos [37].

Acknowledgements We thank Mihir Bellare and Ran Canetti for comments on the model for key usability. We also thank the anonymous referees for their useful comments.

References

[1] IEEE P802.11i/D10.0. Medium Access Control (MAC) security enhancements, amendment 6 to IEEE Standard for local and metropolitan area networks part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications., April 2004.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[3] P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proc. of the 18th IEEE Computer Security Foudnations Workshop*, pages 170–184, 2005.

[4] R. Alur and T. A. Henzinger. Computer-aided verification. an introduction to model building and model checking for concurrent systems. Draft, 1998.

[5] M. Backes, A. Datta, A. Derek, J. C. Mitchell, and M. Turuani. Compositional analysis of contract signing protocols. In *Proceedings of 18th IEEE Computer Security Foundations Workshop*. IEEE, 2005. To appear.

[6] M. Backes and B. Pfitzmann. Limits of the cryptographic realization of XOR. In *Proc. of the 10th European Symposium on Research in Computer Security*. Springer-Verlag, 2005.

[7] M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. In *Proc. IEEE Symposium on Security and Privacy*, pages 171–182. IEEE, 2005.

[8] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015, 2003.

[9] M. Baudet, V. Cortier, and S. Kremer. Computationally Sound Implementations of Equational Theories against Passive Adversaries. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663, Lisboa, Portugal, July 2005. Springer.

[10] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authenticationand key exchange protocols. In *Proc. of the 30th Annual Symposium on the Theory of Computing*, pages 419–428. ACM, 1998.

[11] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '93)*, pages 232–249. Springer-Verlag, 1994.

[12] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of FOCS'01*, pages 136–145, 2001.

[13] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Using probabilistic i/o automata to analyze an oblivious transfer protocol. Technical Report MIT-LCS-TR-1001, MIT CSAIL, 2005.

[14] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proc. of EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474, 2001.

[15] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 337–351, London, UK, 2002. Springer-Verlag.

[16] R. Canetti and T. Rabin. Universal composition with joint state. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281. Springer-Verlag, 2003.

[17] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proceedings of 14th European Symposium on Programming (ESOP'05)*, Lecture Notes in Computer Science, pages 157–171. Springer-Verlag, 2005.

[18] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 109–125. IEEE, 2003.

[19] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 2005.

[20] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP '05)*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[21] T. Dierks and C. Allen. The TLS Protocol — Version 1.0. IETF RFC 2246, January 1999.

[22] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.

[23] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 241–255. IEEE, 2001.

[24] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.

[25] S. O. Ehmety and L. C. Paulson. Program composition in isabelle/unity. In *16th International Parallel and Distributed Processing Symposium (IPDPS 2002), Proceedings*. IEEE Computer Society, 2002.

[26] S. O. Ehmety and L. C. Paulson. Mechanizing compositional reasoning for concurrent systems: some lessons. *Formal Aspects of Computing*, 17(1):58–68, 2005.

[27] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171, Oakland, CA, May 1998. IEEE Computer Society Press.

[28] A. Freier, P. Karlton, and P. Kocher. The SSL protocol version 3.0. IETF Internet draft, November 18 1996.

[29] V. Gligor and D. O. Horvitz. Weak Key Authenticity and the Computational Completeness of Formal Encryption. In D. Boneh, editor, *Advances in cryptology - CRYPTO 2003, proceedings of the 23rd annual international cryptology conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 530–547, Santa Barbara, California, USA, Aug. 2003. Springer-Verlag.

[30] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[31] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[32] P. Gupta and V. Shmatikov. Towards computationally sound symbolic analysis of key exchange protocols. In *Proceedings of ACM Workshop on Formal Methods in Security Engineering*, 2005. to appear.

[33] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, 2005.

[34] R. Impagliazzo and B. Kapron. Logics for reasoning about cryptographic constructions. In *Prof of 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 372–383, 2003.

[35] R. Janvier, L. Mazare, and Y. Lakhnech. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *Proceedings of 14th European Symposium on Programming (ESOP'05)*, Lecture Notes in Computer Science, pages 172–185. Springer-Verlag, 2005.

[36] C. Kauffman. Internet key exchange (IKEv2) protocol. IETF Internet draft, 1994.

[37] J. Kohl and B. Neuman. The Kerberos network authentication service (version 5). IETF RFC 1510, September 1993.

[38] C. Meadows. A model of computation for the NRL protocol analyzer. In *Proceedings of 7th IEEE Computer Security Foundations Workshop*, pages 84–89. IEEE, 1994.

[39] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[40] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.

[41] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, 2004.

[42] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *Proceedings of Seventh USENIX Security Symposium*, pages 201–216, 1998.

[43] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy*, pages 184–200, Washington, 2001.

[44] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley Publishing Co., 2000.

[45] V. Shoup. On formal models for secure key exchange (version 4). Technical Report RZ 3120, IBM Research, 1999.