Inductive Trace Properties for Computational Security

Arnab Roy, Anupam Datta, Ante Derek, John C. Mitchell

Abstract

Protocol authentication properties are generally trace-based, meaning that authentication holds for the protocol if authentication holds for individual traces (runs of the protocol and adversary). Computational secrecy conditions, on the other hand, often are not trace based: the ability to computationally distinguish a system that transmits a secret from one that does not is measured by overall success on the *set* of all traces of each system. This presents a challenge for inductive or compositional methods: induction is a natural way of reasoning about traces of a system, but it does not appear directly applicable to non-trace properties. We therefore investigate the semantic connection between trace properties that could be established by induction and non-trace-based security requirements. Specifically, we prove that a certain trace property implies computational secrecy and authentication properties, assuming the encryption scheme provides chosen ciphertext security and ciphertext integrity. We also prove a similar theorem for computational secrecy assuming Decisional Diffie-Hellman and a chosen plaintext secure encryption scheme.

1 Introduction

In symbolic and computational models of network protocol execution and attack, a protocol defines a set of possible traces (runs). In the computational model, which we consider in this paper, the set of traces is indexed by a security parameter, and there is a probability distribution arising from randomness used in cryptographic actions in the protocol and randomness used by the protocol adversary. Some properties of a protocol are trace properties, meaning that the property can be observed to hold or fail on an individual trace, and the property holds for a set of traces iff it holds for all but a negligible subset. For example, the authentication property "if Bob accepts a key for use with Alice, then Alice participated in the same session of the same key generation protocol" is a trace property. We can see, for any given trace, whether Bob accepted the key and whether Alice participated in the same session of the same protocol. However, many natural secrecy conditions, in the computational model using probabilistic polynomial-time computation, are not trace based. Computational indistinguishability, for example, requires that no computational observer can feasibly distinguish a situation in which a secret is transmitted from a situation in which some non-informative values are transmitted instead. If we look at a single trace, this gives no real information about how likely an observer is to succeed. Instead, we must look at the probability distribution on traces, and determine the probability of success over the entire distribution. The nature of this property presents a challenge for proving computational secrecy properties of protocols, since trace-based properties are naturally amenable to induction, while non-trace-based properties are not. If we assume inductively that a trace-based property holds, this means it holds for (almost) all traces, and we can consider the effect of adding one more step to each trace. If the effect preserves the property on each trace, then we conclude that the property holds for the

protocol as a whole. Since this form of argument only works for trace-based properties, it does not appear applicable to important computational security properties.

In this paper, we develop foundations for inductive proofs of computational security properties by proving connections between selected trace properties and useful non-trace properties. We say that a protocol is *secretive* if the protocol participants protect values intended to be kept secret in certain ways. While one cannot immediately see syntactically whether a protocol is secretive or not, this is a trace property because it only requires certain individual actions by honest parties to a protocol. We prove that all secretive protocols have computational secrecy and authentication properties, assuming the encryption scheme used provides chosen ciphertext security and ciphertext integrity. In addition, we prove a similar theorem for computational secrecy assuming Decisional Diffie-Hellman and a chosen plaintext secure encryption scheme. These results strengthen related results of [8] in that we cover a broader class of protocol properties, allow "nonces" used as keys, and make weaker cryptographic assumptions. The computational security guarantees for secretive protocols are established first for the simple case when secrets are protected by pre-shared "level-0" keys (Theorems 1-3), then generalized (Theorems 4-6) under the condition that each key is protected by predecessor keys in an acyclic graph. This condition avoids the difficulty of dealing with key cycles while assuming only IND-CCA security of encryption. However, the properties we are able to prove do depend on key use, as should be expected in light of previous results [8, 17, 12, 21]. Specifically, we prove strong key indistinguishability properties for a class of secretive protocols that do not use the established secret as a key (Theorems 1, 4), and a weaker key usability property for secretive protocols that allows the established secret to be used as a key (Theorems 2, 5, 7).

While several approaches are possible, we do not present methods for proving that a protocol is secretive. Many protocols have steps that receive a message encrypted with one key, and send some of its parts out encrypted with a different key. For such protocols, one might use an inductive argument about basic receive-send protocol steps, along the lines of the "rank function method" [22] and related work in the strand space approach [23], both previously applied to symbolic execution models. In forthcoming work, we develop a form of secrecy induction general enough to cover Kerberos and a Diffie-Hellman induction general enough to address properties of ISO-9798-3 and IKEv2. An important aspect of the setting we develop in this paper is that we do not need the strong cryptographic assumptions that are needed to show stronger correspondences between symbolic and computational models of protocol execution and attack (see, e.g., [19, 8, 1]). Intuitively, a correspondence between the set of symbolic executions of a protocol, under attack by a symbolic adversary, and the set of computational executions, under attack by a computational adversary, may require cryptographic assumptions that make cryptographic operations as opaque to a computational attacker as they are to a symbolic attacker. In our approach, however, we work directly with the computational model, using traces that contain symbolic actions by the protocol participants (but not by the attacker), and do not require a correspondence between computational and symbolic models.

Section 2 describes the protocol programming language, computational execution model and security properties. A trace-based definition of "secretive protocols" and associated computational security theorems (Theorems 1–6) are presented in section 3. A similar trace-based definition of protocols that use the Diffie-Hellman primitive safely and associated computational secrecy theorem (Theorem 7) is presented in section 4. Conclusions appear in section 5.

2 Computational Model

2.1 Modeling Protocols

A simple protocol programming language is used to represent a *protocol* by a set of programs one for each *role*, such as "Initiator", "Responder" or "Server". A program is a sequence of protocol actions to be executed by an honest participant (see [13, 9, 10] for the syntax and operational semantics of the language). For the purpose of this paper, it is sufficient to know that protocol actions include nonce generation, encryption, decryption, Diffie-Hellman operations and communication steps (sending and receiving). Symmetric encryption with a nonce as a key signifies encryption with a key deterministically generated from the nonce. A principal executing an instance of a role is called a *thread*. A principal can simultaneously execute multiple threads.

We consider a standard two-phase execution model as in [5]. 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 encryption keys 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 [5]. The length of keys, nonces, etc. as well as the running time of the protocol parties and the attacker are polynomially bounded in the security parameter.

Informally, a *trace* is a record of all actions executed by honest principals and the attacker during protocol execution. Since honest principals execute symbolic programs, a trace contains symbolic descriptions of the actions executed by honest parties as well as the mapping of bitstrings to variables. On the other hand, since the attacker is an arbitrary probabilistic poly-time algorithm, the trace only records the send-receive actions of the attacker (and the corresponding mapping to bitstrings), but not her internal actions. More formally, a trace is a pair $\langle e, \lambda \rangle$, where e records the symbolic actions of protocol participants and λ maps symbolic terms in actions to bitstrings using appropriate functions. For example, if the symbolic action involves some thread generating a new nonce s, then $\lambda(s)$ is the bitstring obtained by applying a nonce-generation algorithm (which uses the random coins available to that thread). Similarly, symbolic symmetric encryption terms are mapped to bitstrings obtained by applying an encryption function to the bitstring representation of the corresponding plaintext term given by λ. The computational interpretation of decryption, and Diffie-Hellman actions are defined similarly.

2.2 Modeling Security Properties

Authentication and integrity are generally trace properties. In this paper, we focus on simple integrity properties of the form that a certain encrypted message was produced by a specific principal. Such a property is satisfied by a protocol if for all probabilistic polytime attackers and sufficiently large security parameters this property holds in "almost all" runs of the protocol. Here, the use of the condition "almost all" allows for the fact that the property may not hold in a negligible fraction of the runs as is standard in cryptographic studies [5]. The interested reader is referred to [11] for a formal definition.

Computational secrecy is a more subtle property. It is a property of a set of traces and not a single trace. We consider two notions of computational secrecy—one based on the standard cryptographic notion of *indistinguishability* and the other called *key usability* first presented in [12].

We describe some problems with inductive reasoning about key indistinguishability and discuss the alternative condition that appears more suitable for our purposes.

2.2.1 Key indistinguishability

Key indistinguishability [5, 3] roughly means that an attacker should not be able to distinguish between the actual key produced by the protocol and a random key drawn from the same distribution. This idea is formalized using a standard cryptographic game. The game 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 execution model described in Section 2.1. At the end of the key exchange phase, the adversary selects a challenge session among all sessions executed by the honest parties, and outputs some state information 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. At this point, the experiment enters its second phase—the *challenge phase* where the goal of the adversary $\mathcal{A}_c$ is to distinguish the key k from a random key r drawn from the same distribution using the state information previously output by $\mathcal{A}_e$. The protocol is said to satisfy key indistinguishability if the success probability of $\mathcal{A}_c$ is bounded above $1/2$ by a negligible function of the security parameter.

Key indistinguishability turns out to be too strong a condition in many practical scenarios. Specifically, even if a key exchange protocol run in isolation satisfies this condition, 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 [14]; see [20] 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 [16] does *not* require that the keys used remain indistinguishable from random. To circumvent the technical problems we encountered in working with key indistinguishability, we developed an alternative notion in [12] that is parameterized by the security goal of the application in which the resulting key is used.

Some interesting issues related to key indistinguishability were raised by Rackoff, who proposed an example protocol that is explained in the appendix of [7]. 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 [14]) 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.

2.2.2 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 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 summarized below (see [12] for a complete definition).

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. At the end of the key exchange phase, the adversary selects a challenge session among all sessions executed by the honest parties, and outputs some state information 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. 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 $\mathcal{A}_e$ 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 [15]. We formalize the case when the game defines IND-CPA security. $\mathcal{A}_c$ 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. 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.

3 Secretive Protocols

In this section, we define a trace property of protocols and show that this property implies computational secrecy and integrity. The computational secrecy properties include key indistinguishability and key usability for IND-CCA secure encryption. These results are established first for the simple case when secrets are protected by pre-shared "level-0" keys (Theorems 1-3), then generalized (Theorems 4-6) under the condition that each key is protected by predecessor keys in an acyclic graph. The proofs use standard cryptographic reductions.

Let s and $\mathcal{K}$ be the symbolic representations of a nonce and a set of keys associated with a specific thread in a trace $\langle e, \lambda \rangle$. Let $\Lambda(\mathcal{K}) = \{\lambda(\mathsf{k}) \mid \mathsf{k} \in \mathcal{K}\}$, the set of bitstring representations of keys in $\mathcal{K}$. We start off with the following intuition for the inductive secrecy of s under the keys $\mathcal{K}$ and then proceed to further refine it:

- a thread which generates a new nonce r in e, with $\lambda(\mathsf{r}) = \lambda(\mathsf{s})$, must ensure that r is encrypted with a key k with bitstring representation $\lambda(\mathsf{k}) \in \Lambda(\mathcal{K})$ in any message sent out.
- whenever a thread decrypts a message with a key k with $\lambda(\mathsf{k}) \in \Lambda(\mathcal{K})$ and parses the decryption, it ensures that the results are encrypted with some key k' with $\lambda(\mathsf{k}') \in \Lambda(\mathcal{K})$ in any message sent out.

Definition 3.1 (Good Terms) *An* $(\mathsf{s}, \mathcal{K})$-good *term for an honest principal is either a received term, term of atomic type different from nonce or key, a nonce different from* s*, or constructed in the following ways: pairing* $(\mathsf{s}, \mathcal{K})$*-good terms, unpairing an* $(\mathsf{s}, \mathcal{K})$*-good term, encrypting an* $(\mathsf{s}, \mathcal{K})$*-good term, encrypting any term with a key in* $\mathcal{K}$ *and decrypting good terms with keys not in* $\mathcal{K}$*.*

Definition 3.2 (Secretive Trace) *A trace* $\langle e, \lambda \rangle$ *is an* $(\mathsf{s}, \mathcal{K})$-secretive trace *if every thread belonging to honest principals sends out only* $(\mathsf{s}, \mathcal{K})$-good *terms.*

Definition 3.3 (Secretive Protocol) *A protocol* $\mathcal{Q}$ *is an* $(\mathsf{s}, \mathcal{K})$-secretive protocol *if for all probabilistic poly-time adversaries* $\mathcal{A}$ *and for all sufficiently large security parameters* η*, the probability that a trace* $t(\mathcal{A}, \mathcal{Q}, \eta)$*, generated by the interaction of* $\mathcal{A}$ *with principals following roles of* $\mathcal{Q}$*, is an* $(\mathsf{s}, \mathcal{K})$*-secretive trace is overwhelmingly close to 1, the probability being taken over all adversary and protocol randomness. Formally,*

$$\begin{aligned} &\forall \textit{ PPT adversary } \mathcal{A}.\ \exists \textit{ negligible function } \nu.\ \exists \eta_0.\ \forall \eta \geq \eta_0. \\ &\quad Pr[t(\mathcal{A}, \mathcal{Q}, \eta) \textit{ is } (\mathsf{s}, \mathcal{K})\textit{-secretive }] \geq 1 - \nu(\eta) \end{aligned}$$

A *level-0* key for a protocol execution is an encryption key which is only used as a key but never as part of a payload. We use multi-party security definitions due to Bellare, Boldyreva and Micali [2] applied to symmetric encryption schemes in the following theorems. In [2], IND-CCA2 and the multi-party IND-CCA game are shown to be asymptotically equivalent.

In all the proofs to do with secretive protocols, we implicitly look at the subset of all traces that are secretive among all possible traces. Since the set of non-secretive traces is a negligible subset of all traces, adversary advantages retain the same asymptotic behaviour - negligible advantages remain negligible and non-negligible advantages remain non-negligible.

The general structure of the proofs of the secrecy theorems is by reduction of the appropriate protocol secrecy game to a multi-party IND-CCA game. That is, given protocol adversary $\mathcal{A}$, we construct an adversary $\mathcal{A}'$ against a multi-party IND-CCA challenger which provides $|\mathcal{K}|$-party Left-or-Right encryption oracles $\mathcal{E}_{k_i}(LoR\ (\cdot, \cdot, b))$ parameterized by a challenge bit b and decryption oracles $\mathcal{D}_{k_i}(\cdot)$ for all $k_i \in \mathcal{K}$ (Following [2], $LoR(m_0, m_1, b)$ is a function which returns m_b).

The strategy of $\mathcal{A}'$ is to provide a simulation of the *secretive protocol* to $\mathcal{A}$ by using these oracles such that the capability of $\mathcal{A}$ to break the indistinguishability or key usability of the nonce can be leveraged in some way to guess the challenge bit b of the multi-party IND-CCA challenger. To this end, $\mathcal{A}'$ employs a *bilateral simulator* $\mathcal{S}$ which randomly chooses two bit-strings s_0, s_1 as alternate

representations of the putative secret s and then simulates execution of the protocol to the protocol adversary $\mathcal{A}$ for both the representations.

As with the execution of the actual protocol, $\mathcal{S}$ receives messages and scheduling information from $\mathcal{A}$ and acts according to the roles of the given protocol. The difference from a normal protocol execution is that in computing bitstring representations of terms that involve s, $\mathcal{S}$ does so for both representations of s. We will show that for secretive protocols the representation of s that $\mathcal{A}$ sees is determined by the challenge bit b of the CCA challenger. The operational semantics of the bilateral simulator is formalized in Table 1. We explain the form of the definition using an example.

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{m}'' \quad \mathsf{m} := \mathtt{pair}\ \mathsf{m}', \mathsf{m}'';}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = pair(lv(\mathsf{m}'), lv(\mathsf{m}'')),\ rv(\mathsf{m}) = pair(rv(\mathsf{m}'), rv(\mathsf{m}''))}$$

The notation $\triangleright\ \mathsf{m}$ means that the symbol m has been computationally evaluated by the simulator according to the semantics. The premise of the rule requires that the symbols m and m' have already been evaluated and we are considering the action $\mathsf{m} := \mathtt{pair}\ \mathsf{m}', \mathsf{m}''$ in some protocol thread. The functions lv and rv map a symbol to its bit-string values corresponding to the representations s_0 and s_1 of s respectively. The function *pair* is the actual computational implementation of pairing. The conclusion of the rule states that $lv(\mathsf{m})$ is evaluated by pairing the bit-strings $lv(\mathsf{m}')$ and $lv(\mathsf{m}'')$ and similarly for $rv(\mathsf{m})$. In simulating the protocol to the protocol adversary, the simulator executes each action of the currently scheduled thread following this definition.

We argue informally here that the inductive structure of encryption and decryption allowed in a secretive protocol ensures that the simulator does not stop due to the operational semantics of a send action. Lemma 3.4 states the general case more formally. Suppose m is a term explicitly constructed from s. As $\mathcal{S}$ is simulating a *secretive protocol*, this term is to be encrypted with a key k in $\mathcal{K}$ to construct a message to be sent out to $\mathcal{A}$. So, $\mathcal{S}$ asks the encryption oracle of the $|\mathcal{K}|$-IND-CCA challenger to encrypt $(lv(\mathsf{m}), rv(\mathsf{m}))$ with k. In addition, this pair of bitstrings is recorded and the result of the query is logged in the set qdb_k. If a message construction involves decryption with a key in $\mathcal{K}$, $\mathcal{S}$ first checks whether the term to be decrypted was produced by an encryption oracle by accessing the log qdb_k—if not, then the decryption oracle is invoked; if yes, then $\mathcal{S}$ uses the corresponding encryption query as the decryption. In the second case the encryption query must have been of the form (m_0, m_1). Following the definition of *secretive protocol*, terms constructed from this decryption will be re-encrypted with a key in $\mathcal{K}$ before sending out. Thus we note here that all such replies will be consistent to $\mathcal{A}$ with respect to any choice of b. The situation becomes trickier when encryption or decryption of a term is required with s as the key. In this case $\mathcal{S}$ encrypts or decrypts with s_0. We therefore always have $lv(\mathsf{m}) = rv(\mathsf{m})$ for any message m being sent out. Hence, the simulator will not get stuck due to a send action.

One subtle issue arises when we consider term deconstructors such as unpairings and decryptions, and pattern matching actions. The bilateral simulation for term constructions like pairing and encryption is fairly straightforward. However, to have a consistent simulation we need to ensure that the success of the decontruction and pattern matching actions are independent of the challenge bit b, *i.e.* if the term for $b = 0$ can be unpaired or decrypted then the corresponding operation also succeeds for the term for $b = 1$; if there is a match for $b = 0$ then there should also be a match for $b = 1$.

For this technical reason, we assume that honest parties conform to certain type conventions. These restrictions may be imposed by prefixing the values of each type (nonces, ids, constant strings, pairs, encryptions with key k, etc.) with a tag such as 'constant' or 'encrypted with key-id

$k - id$' that are respected by honest parties executing protocol roles. The adversary may freely modify or spoof tags or produce arbitrary untagged bitrings. It turns out that the type information carried by terms ensures deconstruction and matching consistency in an overwhelming number of traces. This is stated in lemmas 3.6 and 3.7. The proofs proceed by induction over the operational semantics in Table 1.

Lemma 3.4 *If an honest principal in a trace $\langle e, \lambda \rangle$ constructs an $(\mathsf{s}, \mathcal{K})$-good term m, then any bilateral simulator with parameters $s, \mathcal{K}$, executing symbolic actions e produces identical bitstring representations for m on both sides of the simulation,* i.e., *we will have $\triangleright\ \mathsf{m}$ and $lv(\mathsf{m}) = rv(\mathsf{m})$.*

Proof. The proof is by induction on the construction of good terms. The base cases for received messages, data of atomic type different from nonce or key simply follows from the operational semantics of the simulator. For encryption of a good term and decryption with a key not in $\mathcal{K}$, we use the fact that only the $lv()$ of the key is used in the operational semantics for encryption and decryption, hence the result also has equal $lv()$ and $rv()$ values. The case for encryption of any term with a key in $\mathcal{K}$ follows as the operational semantics for this case produces the same value for $lv()$ and $rv()$ in the result. □

Definition 3.5 ($\cong$) *For symbols m, m', we write $\mathsf{m} \cong \mathsf{m}'$ iff $lv(\mathsf{m}) = lv(\mathsf{m}') \wedge rv(\mathsf{m}) = rv(\mathsf{m}')$.*

Lemma 3.6 (Consistent Deconstructions) *If the bitstring representation of the symbol m is a pair on one side of a bilateral simulation then it is a pair on the other side also; similarly for encryption. Formally,*

- *If $\triangleright\ \mathsf{m}$ and $lv(\mathsf{m}) = pair(l_0, l_1)$ for some l_0, l_1, then $rv(\mathsf{m}) = pair(r_0, r_1)$ for some r_0, r_1 and vice versa.*
- *If $\triangleright\ \mathsf{m}, \triangleright\ \mathsf{k}$ and $lv(\mathsf{m}) = enc(l, lv(\mathsf{k}))$ for some l, then $rv(\mathsf{m}) = enc(r, lv(\mathsf{k}))$ for some r and vice versa.*

Proof. The proof is by simultaneous induction on the following stronger propositions:

- If $\triangleright\ \mathsf{m}$ and $lv(\mathsf{m}) = pair(l_0, l_1)$ for some l_0, l_1, then either $lv(\mathsf{m}) = rv(\mathsf{m})$ or there exists m' such that $\mathsf{m} \cong \mathsf{m}'$ and m' is derived by a `pair` action.
- If $\triangleright\ \mathsf{m}$, $\triangleright\ \mathsf{k}$ and $lv(\mathsf{m}) = enc(l, lv(\mathsf{k}))$ for some l, then either $lv(\mathsf{m}) = rv(\mathsf{m})$ or, $\mathsf{k} \notin \mathcal{K}$ and there exists m', k' such that $\mathsf{m} \cong \mathsf{m}', \mathsf{k} \cong \mathsf{k}'$ and m' is derived by an `enc` $\cdot, \mathsf{k}'$ action.
- If $\triangleright\ \mathsf{m}$ and $lv(\mathsf{m})$ is tagged to be of type nonce then either $lv(\mathsf{m}) = rv(\mathsf{m})$ or, $lv(\mathsf{m}) = s_0 \wedge rv(\mathsf{m}) = s_1$.
- In all other cases, if $\triangleright\ \mathsf{m}$ then $lv(\mathsf{m}) = rv(\mathsf{m})$

Case: Rules `STAT, RECV, NEW` – in all these cases, $lv(\mathsf{m}) = rv(\mathsf{m})$, so we are done. Rules `MTCH, SEND` do not generate new values. Rule `NEWS` satisfies $lv(\mathsf{m}) = s_0 \wedge rv(\mathsf{m}) = s_1$.

Case: Rule `PAIR` – resultant value is a pair of two component values and satisfies the proposition that it has been derived by a `pair` action. Similar for rules `ENC`. For rule `EK`, resultant has

$$\frac{\mathsf{m} \text{ in the static list} \quad \mathsf{m} \notin \mathcal{K}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = rv(\mathsf{m}) = getval()}\ \texttt{STAT} \qquad \frac{\texttt{receive m;}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = rv(\mathsf{m}) = recv()}\ \texttt{RECV}$$

$$\frac{\texttt{new n;} \quad \mathsf{n} \neq \mathsf{s}}{\triangleright\ \mathsf{n},\ lv(\mathsf{n}) = rv(\mathsf{n}) = noncegen()}\ \texttt{NEW} \qquad \frac{\texttt{new s;}}{\triangleright\ \mathsf{s},\ lv(\mathsf{n}) = s_0,\ rv(\mathsf{n}) = s_1}\ \texttt{NEWS}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{m}'' \quad \mathsf{m} := \texttt{pair}\ \mathsf{m}', \mathsf{m}'';}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = pair(lv(\mathsf{m}'), lv(\mathsf{m}'')),\ rv(\mathsf{m}) = pair(rv(\mathsf{m}'), rv(\mathsf{m}''))}\ \texttt{PAIR}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{fst}\ \mathsf{m}';}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = fst(lv(\mathsf{m}')),\ rv(\mathsf{m}) = fst(rv(\mathsf{m}'))}\ \texttt{FST}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{snd}\ \mathsf{m}';}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = snd(lv(\mathsf{m}')),\ rv(\mathsf{m}) = snd(rv(\mathsf{m}'))}\ \texttt{SND}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{enc}\ \mathsf{m}', \mathsf{k}; \quad \mathsf{k} \in \mathcal{K}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = rv(\mathsf{m}) = \mathcal{E}_k(lv(\mathsf{m}'), rv(\mathsf{m}')),}\ \texttt{EK}$$

$$qdb_k \leftarrow qdb_k \cup \{lv(\mathsf{m})\}, dec_k^0(lv(\mathsf{m})) = lv(\mathsf{m}'), dec_k^1(lv(\mathsf{m})) = rv(\mathsf{m}')$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \mathsf{m}', \mathsf{k}; \quad \mathsf{k} \in \mathcal{K} \quad lv(\mathsf{m}') \notin qdb_k}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = \mathcal{D}_k(lv(\mathsf{m}'))}\ \texttt{DLK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \mathsf{m}', \mathsf{k}; \quad \mathsf{k} \in \mathcal{K} \quad lv(\mathsf{m}') \in qdb_k}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = dec_k^0(lv(\mathsf{m}'))}\ \texttt{DLKQ}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \mathsf{m}', \mathsf{k}; \quad \mathsf{k} \in \mathcal{K} \quad rv(\mathsf{m}') \notin qdb_k}{\triangleright\ \mathsf{m},\ rv(\mathsf{m}) = \mathcal{D}_k(rv(\mathsf{m}'))}\ \texttt{DRK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \mathsf{m}', \mathsf{k}; \quad \mathsf{k} \in \mathcal{K} \quad rv(\mathsf{m}') \in qdb_k}{\triangleright\ \mathsf{m},\ rv(\mathsf{m}) = dec_k^1(rv(\mathsf{m}'))}\ \texttt{DRKQ}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{enc}\ \mathsf{m}', \mathsf{n}; \quad \mathsf{n} \notin \mathcal{K} \quad \mathsf{n} \text{ tagged nonce or key}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = enc(lv(\mathsf{m}'), keygen(lv(\mathsf{n}))),\ rv(\mathsf{m}) = enc(rv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{ENC}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec}\ \mathsf{m}', \mathsf{n}; \quad \mathsf{n} \notin \mathcal{K} \quad \mathsf{n} \text{ tagged nonce or key}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = dec(lv(\mathsf{m}'), keygen(lv(\mathsf{n}))),\ rv(\mathsf{m}) = dec(rv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{DEC}$$

$$\frac{\triangleright\ \mathsf{m} \quad \triangleright\ \mathsf{m}' \quad \texttt{match m as m}'; \quad (lv(\mathsf{m}) = lv(\mathsf{m}')) \oplus (rv(\mathsf{m}) = rv(\mathsf{m}'))}{\text{stop simulation}}\ \texttt{MTCH}$$

$$\frac{\triangleright\ \mathsf{m} \quad \texttt{send m;} \quad lv(\mathsf{m}) \neq rv(\mathsf{m})}{\text{stop simulation}}\ \texttt{SEND}$$

Table 1: Operational Semantics of the Simulator with Parameters $\mathsf{s}, \mathcal{K}$

the same lv and rv values.

Case: $\triangleright\ \mathsf{m}',\ \mathsf{m} := \mathtt{fst}\ \mathsf{m}'$; Now, $lv(\mathsf{m}') = pair(lv(\mathsf{m}), l)$ for some l. Therefore, by IH, either $lv(\mathsf{m}') = rv(\mathsf{m}')$ or m' was derived by a `pair` action.

If $lv(\mathsf{m}') = rv(\mathsf{m}')$, then $lv(\mathsf{m}) = fst(lv(\mathsf{m}')) = fst(rv(\mathsf{m}')) = rv(\mathsf{m})$ and we are done.

If $\mathsf{m}'' \cong \mathsf{m}'$ was derived by a `pair` action, say: $\triangleright\ \mu, \triangleright\ \mu', \mathsf{m}'' := \mathtt{pair}\ \mu, \mu'$; Therefore,

$$lv(\mathsf{m}'') = lv(\mathsf{m}') \Rightarrow fst(lv(\mathsf{m}'')) = fst(lv(\mathsf{m}')) \Rightarrow lv(\mu) = lv(\mathsf{m})$$

$$rv(\mathsf{m}'') = rv(\mathsf{m}') \Rightarrow fst(rv(\mathsf{m}'')) = fst(rv(\mathsf{m}')) \Rightarrow rv(\mu) = rv(\mathsf{m})$$

Therefore $\mu \cong \mathsf{m}$ and we are through by IH. The induction over rule `SND` is similar.

Case: $\triangleright\ \mathsf{m}', \triangleright\ \mathsf{k},\ \mathsf{m} := \mathtt{dec}\ \mathsf{m}', \mathsf{k}$; Now, $lv(\mathsf{m}') = enc(lv(\mathsf{m}), lv(\mathsf{k}))$. Therefore, by IH, either $lv(\mathsf{m}') = rv(\mathsf{m}')$ or, $\mathsf{k} \notin \mathcal{K}$ and some m'' was derived by an $\mathtt{enc}\ \cdot, \mathsf{k}''$ action, with $\mathsf{m}'' \cong \mathsf{m}', \mathsf{k}'' \cong \mathsf{k}'$.

If $lv(\mathsf{m}') = rv(\mathsf{m}')$ and their value is $\in qdb_k$ then rule `EK` must have been invoked for an $\mathsf{m}'' \cong \mathsf{m}'$ at some point before, when the value was entered in the qdb_k - i.e. we had $\triangleright\ \mu,\ \mathsf{m}'' := \mathtt{enc}\ \mu, \mathsf{k};\ \mathsf{k} \in \mathcal{K}$. Therefore $lv(\mu) = dec_k^0(lv(\mathsf{m}'')) = dec_k^0(lv(\mathsf{m}')) = lv(\mathsf{m})$, by `DLKQ`, and $rv(\mu) = dec_k^1(rv(\mathsf{m}'')) = dec_k^1(rv(\mathsf{m}')) = rv(\mathsf{m})$, by `DRKQ`. Hence, $\mu \cong \mathsf{m}$ and the induction follows by IH.

If $lv(\mathsf{m}') = rv(\mathsf{m}')$ and their value is $\notin qdb_k$, then $enc(lv(\mathsf{m}), lv(k)) = enc(rv(\mathsf{m}), lv(k))$. Therefore $lv(\mathsf{m}) = rv(\mathsf{m})$.

If $lv(\mathsf{m}') \neq rv(\mathsf{m}')$, then $\mathsf{k} \notin \mathcal{K}$ and some m'' was derived by an $\mathtt{enc}\ \cdot, \mathsf{k}''$ action, with $\mathsf{m}'' \cong \mathsf{m}', \mathsf{k}'' \cong \mathsf{k}'$. Since $lv(\mathsf{m}') \neq rv(\mathsf{m}')$, this derivation must have been by the `ENC` rule, say: $\triangleright\ \mu, \triangleright\ \mathsf{k}'', \mathsf{m}'' := \mathtt{enc}\ \mu, \mathsf{k}''$. Therefore,

$$lv(\mathsf{m}'') = lv(\mathsf{m}') \Rightarrow enc(lv(\mu), lv(\mathsf{k}'')) = enc(lv(\mathsf{m}), lv(\mathsf{k})) \Rightarrow lv(\mu) = lv(\mathsf{m})$$

$$rv(\mathsf{m}'') = rv(\mathsf{m}') \Rightarrow enc(rv(\mu), lv(\mathsf{k}'')) = enc(rv(\mathsf{m}), lv(\mathsf{k})) \Rightarrow rv(\mu) = rv(\mathsf{m})$$

Therefore $\mu \cong \mathsf{m}$ and we are through by IH.

Lemma 3.7 (Consistent Matchings) *If the symbols m and m' are computationally evaluated by a bilateral simulator, i.e., $\triangleright\ \mathsf{m}, \triangleright\ \mathsf{m}'$ holds, and the action $\mathtt{match}\ \mathsf{m}\ \mathtt{as}\ \mathsf{m}'$ is executed, then with overwhelming probability, the match succeeds on the left side iff it succeeds on the right.*

Proof. Suppose $\triangleright\ \mathsf{m}, \triangleright\ \mathsf{m}'$ holds, and the action `match m as m'` is executed and the match succeeds exactly on one side. Therefore, for at least one of m, m'—say m'—it must be that $lv(\mathsf{m}') \neq rv(\mathsf{m}')$. Therefore, m' must have been derived by either pairing or, an encryption with a key k such that $\triangleright\ \mathsf{k}, \mathsf{k} \notin \mathcal{K}$, or, $\mathsf{m}' \cong \mathsf{s}$.

- Suppose $\mathsf{m}'', \mathsf{m}'' \cong \mathsf{m}'$, was derived by a `pair` action – $\triangleright\ \mu, \triangleright\ \mu', \mathsf{m}'' = \mathtt{pair}\ \mu, \mu'$. It must be that for one of μ, μ', say μ, $lv(\mu) \neq rv(\mu)$.
- Suppose $\mathsf{m}'', \mathsf{m}'' \cong \mathsf{m}'$, was derived by a `enc` action – $\triangleright\ \mu, \triangleright\ \mathsf{k}, \mathsf{k} \notin \mathcal{K}, \mathsf{m}'' = \mathtt{enc}\ \mu, \mathsf{k}$. It must be that $lv(\mu) \neq rv(\mu)$.

Now the argument about m' can be reapplied to μ since $lv(\mu) \neq rv(\mu)$. It therefore follows that as we iteratively look at the derivations, we will eventually reach a symbol n, such that $\mathsf{n} \cong \mathsf{s}$. Intuitively what this means is that it is possible to obtain s from m' through a series of unpairings

and decryptions using keys that honest principals have used. We will be guided by this intuition to build an adversary against a $|\mathcal{K}|$-IND-CCA challenger.

The $|\mathcal{K}|$-IND-CCA adversary we build consists of a modified simulator $\mathcal{S}'$ interacting with a protocol adversary $\mathcal{A}'$ which can instruct $\mathcal{S}'$ to undertake individual actions, represented symbollically. As compared to $\mathcal{S}$, $\mathcal{S}'$ follows the same operational semantics on individual symbolic actions from the given protocol syntax but instead of conforming to a specific protocol role, these actions are dictated by the adversary. In addition $\mathcal{S}'$ outputs status of actions such as 'match succeeded or failed on left, right or both sides', 'unpairing succeeded/failed' and 'decryption succeeded/failed', 'send failed as $lv() \neq rv()$'.

The algorithm $\mathcal{A}'$ first uses $\mathcal{S}'$ to simulate the protocol execution to the protocol adversary $\mathcal{A}$. Whenever $\triangleright\, \mathsf{m}, \triangleright\, \mathsf{m}'$ holds, and the action `match m as m'` is executed and the match succeeds exactly on one side, the simulator $\mathcal{S}'$ tells $\mathcal{A}'$ so. At this point, $\mathcal{A}'$ executes the function `parse-get-s` (Table 2) on the symbols m and m'. We will show that for at least one of m and m', `parse-get-s`() will output a (b', s') such that $s' = s_{b'}$. Also with overwhelming probability $b' = b$ will hold, where b is the $|\mathcal{K}|$-IND-CCA challenger bit.

Suppose the matching succeeds on the left side, i.e., $lv(\mathsf{m}) = lv(\mathsf{m}')$ but $rv(\mathsf{m}) \neq rv(\mathsf{m}')$. Since $lv(\mathsf{m}) = lv(\mathsf{m}')$, note that m and m' will have an identical parsing sequence. Since also $lv(\mathsf{m}') \neq rv(\mathsf{m}')$, the parsing should reach, at least once, an n' with $\mathsf{n}' \cong \mathsf{s}$. The parse tree of m should also have a corresponding n with $lv(\mathsf{n}) = s_0$. Since $rv(\mathsf{m}) \neq rv(\mathsf{m}')$, for at least one of these leaves n we should have $rv(\mathsf{n}) \neq s_1$, but in this case we can only have $rv(\mathsf{n}) = s_0 = lv(\mathsf{n})$. Therefore `send n` action is allowed for $\mathcal{S}'$. Therefore `parse-get-s`(m) succeeds and outputs $(0, s_0)$.

Case 1: matching succeeds only in the left. We argue in this case that $b = 0$ with overwhelming probability. Suppose on the contrary, $b = 1$. Then the protocol simulation from $\mathcal{A}'$'s perspective used s_0 only as a key to an IND-CCA secure encryption scheme, if used at all. Since s_0 was chosen randomly, the probability of an adversary coming up with s_0 is negligible. Hence $b = 1$ only with negligible probability.

Case 2: matching succeeds only in the right. We argue in this case that $b = 1$ with overwhelming probability. Suppose on the contrary, $b = 0$. Then the protocol simulation from $\mathcal{A}'$'s perspective never used s_1. Since s_1 was chosen randomly, the probability of an adversary to come up with s_1 is negligible. Hence $b = 0$ only with negligible probability. □

Theorem 3.8 (CCA security - No keying - level 1) *Assume that a probabilistic poly-time adversary interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. Also assume that s is never used as a key by the honest principals. The adversary has negligible advantage at distinguishing the value of s from random, after the interaction if the encryption scheme is IND-CCA secure. In other words, the protocol satisfies key indistinguishability for s.*

Proof. Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. We will show that if $\mathcal{A}$ has non-negligible advantage at distinguishing the value of s from random, after the interaction, then we can construct a $|\mathcal{K}|$-IND-CCA adversary $\mathcal{A}_1$ with non-negligible advantage against the encryption scheme.

Adversary $\mathcal{A}_1$ employs a bilateral simulator $\mathcal{S}$ which randomly chooses two bit-strings s_0, s_1 as alternate representations of the putative secret s and then simulates execution of the protocol to the protocol adversary $\mathcal{A}$ for both the representations. Since, s is never used as a key, we note here that all such replies will be consistent to $\mathcal{A}$ with respect to any choice of b.

Algorithm `parse-get-s`(m)

```
if (S' : m' := fst m;) succeeds then      [m' is a new symbol]
  r ← parse-get-s(m')
  if (r ≠ ⊥) return r
  else S' : m'' := snd m;      [m'' is a new symbol]
  return parse-get-s(m'')
else do
  for all k, such that ▷ k and k ∉ K
    if(S' : m' := symdec m, k;) succeeds then      [m' is a new symbol]
      return parse-get-s(m')
else do
  if(match m as s) succeeds on exactly one side
    m ← result(S' : send m;)
    if match succeeded on left side then b ← 0 else b ← 1
    return (b, m)
  else return ⊥
else return ⊥
```

Table 2: Algorithm parse-get-s

In the second phase, $\mathcal{A}_1$ chooses a bit d' and sends $s_{d'}$ to $\mathcal{A}$. If $\mathcal{A}$ replies that this is the actual nonce used, then $\mathcal{A}_1$ finishes by outputting $d = d'$, otherwise it outputs $d = \bar{d'}$ and finishes. The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}|$-IND-CCA challenger is:

$$\mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{1}$$

Since $\mathcal{A}$ has a non-negligible advantage at distinguishing s from random, the quantity on the RHS must be non-negligible. Therefore the advantage in the LHS must be non-negligible and hence we are done. □

Theorem 3.9 (CCA security - Keying - level 1) *Assume that a probabilistic poly-time adversary interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. Honest principals are allowed to use s as a key. The adversary has negligible advantage at winning an IND-CCA game against a symmetric encryption challenger, using the key s, after the interaction if the encryption scheme is IND-CCA secure. In other words, the protocol satisfies IND-CCA key usability for s.*

Proof. Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. We will show that if $\mathcal{A}$ has non-negligible advantage at winning an IND-CCA game against a symmetric encryption challenger, using the key s, after the interaction then we can construct either a $|\mathcal{K}|$-IND-CCA adversary $\mathcal{A}_1$ or an IND-CCA adversary $\mathcal{A}_2$ with non-negligible advantages against the encryption scheme.

We proceed as in the proof of theorem 3.8 to construct the adversary $\mathcal{A}_1$. The situation becomes different when encryption or decryption of a term is required with s as the key. In this case $\mathcal{S}$ encrypts or decrypts with s_0.

In the second phase, $\mathcal{A}_1$ uniformly randomly chooses a bit b' and provides oracles $\mathcal{E}_{s_0}(LoR(\cdot, \cdot, b'))$ and $\mathcal{D}_{s_0}(\cdot)$ to $\mathcal{A}$ for an IND-CCA game. $\mathcal{A}$ finishes by outputting a bit d'. If $b' = d'$, $\mathcal{A}_1$ outputs $d = 0$ else outputs $d = 1$. The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}|$-IND-CCA challenger is:

$$\mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{2}$$

Observe that if $b = 0$ then s was consistently represented by s_0 in messages sent to $\mathcal{A}$. Hence, the first probability is precisely the probability of $\mathcal{A}$ winning an IND-CCA challenge with s as the key after interacting with an $(\mathsf{s}, \mathcal{K})$-*secretive protocol*. We will now bound the second probability. We start by constructing a second adversary $\mathcal{A}_2$ which interacts with an alternate simulator $\mathcal{S}'$ (described in Table 3) which has all the keys in $\mathcal{K}$, randomly generates a nonce s_1 and has access to an encryption oracle $\mathcal{E}_{s_0}(LoR(\cdot, \cdot, b_1))$ and a decryption oracle $\mathcal{D}_{s_0}(\cdot)$. It has a similar behaviour towards $\mathcal{A}$ as $\mathcal{S}$ had except that when constructing terms with s, it uses s_1 but when required to encrypt or decrypt using s, it queries $\mathcal{E}_{s_0}(LoR(\cdot, \cdot, b_1))$ or $\mathcal{D}_{s_0}(\cdot)$. In the second phase, $\mathcal{A}_2$ uses the oracles $\mathcal{E}_{s_0}(LoR(\cdot, \cdot, b_1))$ and $\mathcal{D}_{s_0}(\cdot)$ to provide the IND-CCA challenger to $\mathcal{A}$. $\mathcal{A}$ finishes by outputting a bit d_1. $\mathcal{A}_2$ outputs d_1. We observe here that if $b = 1$ for the earlier LoR oracle, it makes no difference to the algorithm $\mathcal{A}$ whether it is interacting with $\mathcal{A}_1$ or $\mathcal{A}_2$. Thus we have:

$$(1/2)\mathbf{Adv}_{IND-CCA,\mathcal{A}_2}(\eta) = Pr[d_1 = b_1] - 1/2 = Pr[d = 0|b = 1] - 1/2 \tag{3}$$

By the equations 2 and 3 we have:

$$Pr[d = 0|b = 0] - 1/2 = \mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) + (1/2)\mathbf{Adv}_{IND-CCA,\mathcal{A}_2}(\eta)$$

$$\frac{\mathsf{m} \text{ in the static list}}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = getval()}\ \texttt{STAT} \qquad \frac{\texttt{receive m;}}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = recv()}\ \texttt{RECV}$$

$$\frac{\texttt{new n;} \quad \mathsf{n} \neq \mathsf{s}}{\triangleright\ \mathsf{n},\ v(\mathsf{n}) = noncegen()}\ \texttt{NEW} \qquad \frac{\texttt{new s;}}{\triangleright\ \mathsf{s},\ v(\mathsf{n}) = s_1}\ \texttt{NEWS}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{m}'' \quad \mathsf{m} := \texttt{pair m}', \mathsf{m}'';}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = pair(v(\mathsf{m}'), v(\mathsf{m}''))}\ \texttt{PAIR}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{fst m}';}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = fst(v(\mathsf{m}'))}\ \texttt{FST}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{snd m}';}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = snd(v(\mathsf{m}'))}\ \texttt{SND}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{k} \quad \mathsf{m} := \texttt{enc m}', \mathsf{k}; \quad v(\mathsf{k}) \neq s_1}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = enc(v(\mathsf{m}'), v(\mathsf{k}))}\ \texttt{ENC}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{k} \quad \mathsf{m} := \texttt{dec m}', \mathsf{k}; \quad v(\mathsf{k}) \neq s_1}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = dec(v(\mathsf{m}'), v(\mathsf{k}))}\ \texttt{DEC}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{enc m}', \mathsf{n}; \quad v(\mathsf{n}) = s_1}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = \mathcal{E}_{s_0}(v(\mathsf{m}'), v(\mathsf{m}')),}\ \texttt{EK}$$
$$qdb \leftarrow qdb \cup \{v(\mathsf{m})\}, dec_{s_0}(v(\mathsf{m})) = v(\mathsf{m}')$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec m}', \mathsf{n}; \quad v(\mathsf{n}) = s_1 \quad v(\mathsf{m}) \notin qdb}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = \mathcal{D}_{s_0}(v(\mathsf{m}'))}\ \texttt{DK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec m}', \mathsf{n}; \quad v(\mathsf{n}) = s_1 \quad v(\mathsf{m}) \in qdb}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = dec_{s_0}(v(\mathsf{m}'))}\ \texttt{DKQ}$$

$$\frac{\triangleright\ \mathsf{m} \quad \triangleright\ \mathsf{m}' \quad \texttt{match m as m}';}{\text{stop thread if } v(\mathsf{m}) \neq v(\mathsf{m}')\text{; else continue}}\ \texttt{MTCH}$$

$$\frac{\triangleright\ \mathsf{m} \quad \texttt{send m;}}{send(v(\mathsf{m}))}\ \texttt{SEND}$$

Table 3: Operational Semantics of the Alternate Simulator with Parameters $\mathsf{s}, \mathcal{K}$ in the case $b = 1$

As the probablity in the LHS is non-negligible, at least one of the advantages in the RHS must be non-negligible and hence we are done. □

If a protocol is a *secretive protocol* with respect to nonce k and set of level-0 keys $\mathcal{K}$ then we will call k a level-1 key for the protocol, protected by $\mathcal{K}$. Now we state a theorem establishing the integrity of encryptions done with level-1 keys. The security definition INT-CTXT for ciphertext integrity is due to [4] and also referred to as *existential unforgeability* of ciphertexts in [18].

Theorem 3.10 (CTXT integrity - level 1) *Assume that a probabilistic poly-time adversary interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. During the protocol run, if an honest principal decrypts a ciphertext with key s successfully, then with overwhelming probability the ciphertext was produced by an honest principal by encryption with s if the encryption scheme is IND-CCA and INT-CTXT secure.*

Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with $\mathcal{K}$ consisting of level-0 keys only. Suppose during the protocol run, an honest party decrypts a ciphertext with key s successfully which was not produced by an honest party by encryption with s. We build a $|\mathcal{K}|$-IND-CCA adversary $\mathcal{A}_1$ against set of keys $\mathcal{K}$ in the lines of the proof of theorem 3.9. However, this new $\mathcal{A}_1$ computes d in a different way. Recall that $\mathcal{S}$ uses s_0 when it intends to encrypt or decrypt using s. In the course of interaction with $\mathcal{A}_1$, if $\mathcal{S}$ succeeds in decrypting a ciphertext with key s_0 which was not produced at a previous stage by $\mathcal{S}$ by encryption with s_0, $\mathcal{A}_1$ outputs $d = 0$. Otherwise, it outputs $d = 1$. The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}|$-IND-CCA challenger is:

$$\mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{4}$$

Now, $Pr[d = 0|b = 0]$ is the probability of $\mathcal{S}$ succeeding in decrypting a ciphertext with key s which was not obtained through encryption by $\mathcal{S}$. $Pr[d = 0|b = 1]$ is the probability of $\mathcal{A}_1$ succeeding in decrypting a ciphertext with level-0 key s_0 (as in this case s_0 was only used as a key). Therefore, using a similar idea as proof of theorem 3.9 we can build an INT-CTXT adversary $\mathcal{A}_2$ against s_0. Therefore,

$$Pr[d = 0|b = 0] = \mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) + \mathbf{Adv}_{INT-CTXT,\mathcal{A}_2}(\eta)$$

As the encryption scheme is both IND-CCA and INT-CTXT secure, both the probabilities on the RHS must be negligible and hence the theorem. □

We now extend theorems 3.8–3.10 to directed key hierarchies. This extension is motivated by the fact that many key distribution protocols (e.g. Kerberos) have key hierarchies with keys protected by lower level keys in the hierarchy.

Definition 3.11 (Key Graph) *Let $\mathcal{K}$ be the symbolic representations of specific nonces and keys in a trace $\langle e, \lambda \rangle$. The* key graph *of $\mathcal{K}$ in a protocol is a directed graph with keys in $\mathcal{K}$ as vertices. There is an edge from key k_1 to k_2 if the protocol is secretive with respect to k_2 and a key set which includes k_1.*

Definition 3.12 (Key Level) *Suppose the key graph of a key set $\mathcal{K}$ in a protocol is directed acyclic. Keys at the root are level 0 keys. The level of any other key is one more than the maximum level among its immediate predecessors.*

Definition 3.13 (Key Basis) *Suppose the key graph of a key set $\mathcal{K}$ in a protocol is directed acyclic. We define its basis $\mathcal{B}(\mathcal{K})$ to be the union of sets of keys at the root which are predecessors of each key in $\mathcal{K}$.*

Theorem 3.14 (CCA security - No Keying) *Assume that a probabilistic poly-time adversary interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with the key graph of $\mathcal{K}$ being a DAG. Also assume that s is never used as a key by the honest principals. The adversary has negligible advantage at distinguishing s from random, after the interaction, if the encryption scheme is IND-CCA secure. In other words, the protocol satisfies key indistinguishability for s.*

Proof. We will prove this by induction over the maximum level of the DAG of $\mathcal{K}$. If $\mathcal{K}$ consists only of level 0 keys then the result follows from theorem 3.8. Suppose the maximum level in the DAG of $\mathcal{K}$ is $(n+1)$ and assume that the theorem holds for maximum level n. Let $\mathcal{K}'$ be the basis $\mathcal{B}(\mathcal{K})$ of the set of keys $\mathcal{K}$.

Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol. We will show that if $\mathcal{A}$ has non-negligible advantage at s from a random bitstring of the same length, after the interaction, then we can construct either a $|\mathcal{K}'|$-IND-CCA adversary $\mathcal{A}_1$ to the encryption scheme or contradict the induction hypothesis.

We will construct an adversary $\mathcal{A}_1$ which has access to a modified bilateral simulator $\mathcal{S}$ (described in Table 4) which, in turn, has access to multi-party LoR encryption oracles $\mathcal{E}_{k_i}(LoR(\cdot,\cdot,b))$ and decryption oracles $\mathcal{D}_{k_i}(\cdot)$ for all $k_i \in \mathcal{K}'$ parameterized by a bit b chosen uniformly randomly. For keys s^i of level ≥ 0, $\mathcal{S}$ chooses random values s_0^i, s_1^i and for s, $\mathcal{S}$ chooses random values s_0, s_1. Intuitively, $\mathcal{S}$ constructs messages to be sent to $\mathcal{A}$ as follows:

- to encrypt the term $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with $\mathsf{k}_i \in \mathcal{K}'$, use response to oracle query $\mathcal{E}_{k_i}(f(s_0, s_0^1, s_0^2, ...), f(s_1, s_1^1, s_1^2, ...), b)$.
- to encrypt $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with s^i, use $\mathcal{E}_{s_0^i}(f(s_0, s_0^1, s_0^2, ...))$.

Decryption operations are served analogously.

In the second phase, $\mathcal{A}_1$ chooses a bit d' and sends $s_{d'}$ to $\mathcal{A}$. If $\mathcal{A}$ replies that this is the actual nonce used, then $\mathcal{A}_1$ finishes by outputting $d = d'$, otherwise it outputs $d = \bar{d'}$ and finishes. The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}|$-IND-CCA challenger is:

$$\begin{aligned}\mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) &= Pr[d=0|b=0] - Pr[d=0|b=1] \\ &= (Pr[d=0|b=0] - 1/2) + (Pr[d=1|b=1] - 1/2) \qquad (5)\end{aligned}$$

The first probability in the RHS is precisely the probability of $\mathcal{A}$ breaking the indistinguishability of s_0 or equivalently of s. In the case when $b = 1$, the terms were constructed in the following manner:

- encrypt $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with $\mathsf{k}_i \in \mathcal{K}'$: $\mathcal{E}_{k_i}(f(s_1, s_1^1, s_1^2, ...))$.
- encrypt $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with s^i: $\mathcal{E}_{s_0^i}(f(s_0, s_0^1, s_0^2, ...))$.

We observe here that $\mathcal{S}$ simulated the execution of another secretive protocol $\mathcal{G}'$ with keys of level $\leq n$ - $s_0^1, s_0^2, ...$ protecting s_0 (operational semantics described in Table 5). This is because the

root level keys no longer protect the other keys in the DAG - we obtain a transformed DAG with the roots of the earlier DAG removed, and hence of maximum level one less. Therefore, we have:

$$Pr[d = 1|b = 1] - 1/2 = (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta) \tag{6}$$

By the equations 5 and 6 we have:

$$Pr[d = 0|b = 0] - 1/2 = \mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) - (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta)$$

As the probablity in the LHS is non-negligible, at least one of the advantages in the RHS must be non-negligible and hence we are done. □

Theorem 3.15 (CCA security - Keying) *Assume that a probabilistic poly-time adversary interacts with an* $(\mathsf{s}, \mathcal{K})$*-secretive protocol with the key graph of* $\mathcal{K}$ *being a DAG. Honest principals are allowed to use* s *as a key. The adversary has negligible advantage at winning an IND-CCA game against a symmetric encryption challenger, using the key* s*, after the interaction if the encryption scheme is IND-CCA secure. In other words, the protocol satisfies IND-CCA key usability for* s*.*

Proof. We will again prove this by induction over the maximum level of the DAG of $\mathcal{K}$. If $\mathcal{K}$ consists only of level 0 keys then the result follows from theorem 3.9. Suppose the maximum level in the DAG of $\mathcal{K}$ is $(n+1)$ and assume that the theorem holds for maximum level n. Let $\mathcal{K}'$ be the basis $\mathcal{B}(\mathcal{K})$ of the set of keys $\mathcal{K}$.

Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol. We will show that if $\mathcal{A}$ has non-negligible advantage at winning an IND-CCA game against a symmetric encryption challenger, using the key s, after the interaction then we can construct either a $|\mathcal{K}'|$-IND-CCA adversary $\mathcal{A}_1$ or contradict the induction hypothesis.

We proceed as in the proof of theorem 3.14 to construct the adversary $\mathcal{A}_1$. The only additional operation is that to encrypt or decrypt the term m with s, $\mathcal{S}$ uses s_0 as the key.

In the second phase, $\mathcal{A}_1$ randomly chooses a bit $b' \leftarrow \{0, 1\}$. $\mathcal{A}$ sends pairs of messages m_0, m_1 to $\mathcal{A}_1$. $\mathcal{A}_1$ replies with $\mathcal{E}_{s_0}(m_{b'})$. Decryption requests are also served by decrypting with key s_0 ciphertexts not obtained by a query in this phase. $\mathcal{A}$ finishes by outputting a bit d'. If $b' = d'$, $\mathcal{A}_1$ outputs $d = 0$ else outputs $d = 1$.

The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}'|$-IND-CCA challenger is:

$$\mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{7}$$

The first probability is precisely the probability of $\mathcal{A}$ breaking the 'good-key'-ness of s_0 or equivalently of s. In the case when $b = 1$, the terms were constructed in the following manner:

- encrypt $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with $\mathsf{k}_i \in \mathcal{K}'$: $\mathcal{E}_{k_i}(f(s_1, s_1^1, s_1^2, ...))$.
- encrypt $f(\mathsf{s}, \mathsf{s}^1, \mathsf{s}^2, ...)$ with s^i: $\mathcal{E}_{s_0^i}(f(s_0, s_0^1, s_0^2, ...))$.
- encrypt term m with s: $\mathcal{E}_{s_0}(m)$.

We observe here that $\mathcal{S}$ simulated the execution of another secretive protocol $\mathcal{G}'$ with keys of level $\leq n$ - $s_0^1, s_0^2, ...$ protecting s_0. This is because the root level keys no longer protect the other

$$\frac{\texttt{new n};\quad \mathsf{n} \notin \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{n},\ lv(\mathsf{n}) = rv(\mathsf{n}) = noncegen()}\ \texttt{KD-NEW}$$

$$\frac{\texttt{new n};\quad \mathsf{n} \in \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{n},\ lv(\mathsf{n}) = noncegen(),\ rv(\mathsf{n}) = noncegen()}\ \texttt{KD-NEWS}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{enc}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \in \mathcal{B}(\mathcal{K})}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = rv(\mathsf{m}) = \mathcal{E}_k(lv(\mathsf{m}'), rv(\mathsf{m}')),}\ \texttt{KD-EK}$$
$$qdb_k \leftarrow qdb_k \cup \{lv(\mathsf{m})\}, dec^0_k(lv(\mathsf{m})) = lv(\mathsf{m}'), dec^1_k(lv(\mathsf{m})) = rv(\mathsf{m}')$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \in \mathcal{B}(\mathcal{K}) \quad lv(\mathsf{m}') \notin qdb_k}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = \mathcal{D}_k(lv(\mathsf{m}'))}\ \texttt{KD-DLK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \in \mathcal{B}(\mathcal{K}) \quad lv(\mathsf{m}') \in qdb_k}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = dec^0_k(lv(\mathsf{m}'))}\ \texttt{KD-DLKQ}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \in \mathcal{B}(\mathcal{K}) \quad rv(\mathsf{m}') \notin qdb_k}{\triangleright\ \mathsf{m},\ rv(\mathsf{m}) = \mathcal{D}_k(rv(\mathsf{m}'))}\ \texttt{KD-DRK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \in \mathcal{B}(\mathcal{K}) \quad rv(\mathsf{m}') \in qdb_k}{\triangleright\ \mathsf{m},\ rv(\mathsf{m}) = dec^1_k(rv(\mathsf{m}'))}\ \texttt{KD-DRKQ}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{enc}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \in \mathcal{K} - \mathcal{B}(\mathcal{K})}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = rv(\mathsf{m}) = enc(lv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{KD-ENC}\quad (*)$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \in \mathcal{K} - \mathcal{B}(\mathcal{K})}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = dec(lv(\mathsf{m}'), keygen(lv(\mathsf{n}))),\ rv(\mathsf{m}) = dec(rv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{KD-DEC}\quad (*)$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{enc}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \notin \mathcal{K}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = enc(lv(\mathsf{m}'), keygen(lv(\mathsf{n}))),\ rv(\mathsf{m}) = enc(rv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{KD-ENC}\quad (*)$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \notin \mathcal{K}}{\triangleright\ \mathsf{m},\ lv(\mathsf{m}) = dec(lv(\mathsf{m}'), keygen(lv(\mathsf{n}))),\ rv(\mathsf{m}) = dec(rv(\mathsf{m}'), keygen(lv(\mathsf{n})))}\ \texttt{KD-DEC}\quad (*)$$

(*) Note that we are only using $lv(\mathsf{n})$ as the key here.

Table 4: Modification of the Operational Semantics with Parameters $\mathsf{s}, \mathcal{K}$ for Key DAGs

$$\frac{\texttt{new n};\quad \mathsf{n} \notin \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{n},\ v(\mathsf{n}) = noncegen()}\ \texttt{KD-NEW}$$

$$\frac{\texttt{new n};\quad \mathsf{n} \in \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{n},\ v(\mathsf{n}) = n_1}\ \texttt{KD-NEWS}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{enc}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \notin \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = enc(v(\mathsf{m}'), v(\mathsf{k}))}\ \texttt{KD-EK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{k};\quad \mathsf{k} \notin \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = dec(v(\mathsf{m}'), v(\mathsf{k}))}\ \texttt{KD-DK}$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{enc}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \in \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = enc(v(\mathsf{m}'), n_0)}\ \texttt{KD-ENC}\quad (*)$$

$$\frac{\triangleright\ \mathsf{m}' \quad \triangleright\ \mathsf{n} \quad \mathsf{m} := \texttt{dec}\ \ \mathsf{m}', \mathsf{n};\quad \mathsf{n} \in \{\mathsf{s}\} \cup (\mathcal{K} - \mathcal{B}(\mathcal{K}))}{\triangleright\ \mathsf{m},\ v(\mathsf{m}) = dec(v(\mathsf{m}'), n_0)}\ \texttt{KD-DEC}\quad (*)$$

Table 5: Operational Semantics of Alternate Simulator for b = 1 with Parameters $\mathsf{s}, \mathcal{K}$ for Key DAGs

keys in the DAG - we obtain a transformed DAG with the roots of the earlier DAG removed, and hence of maximum level one less. Therefore, we have:

$$Pr[d = 0|b = 1] - 1/2 = (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta) \tag{8}$$

By the equations 7 and 8 we have:

$$Pr[d = 0|b = 0] - 1/2 = \mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) + (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta)$$

As the probablity in the LHS is non-negligible, at least one of the advantages in the RHS must be non-negligible and hence we are done. □

Theorem 3.16 (CTXT integrity) *Assume that a probabilistic poly-time adversary interacts with an $(\mathsf{s}, \mathcal{K})$-secretive protocol with the key graph of $\mathcal{K}$ being a DAG. During the protocol run, if an honest principal decrypts a ciphertext with key s successfully, then with overwhelming probability the ciphertext was produced by an honest principal by encryption with s if the encryption scheme is IND-CCA and INT-CTXT secure.*

We will prove this by induction over the maximum level of the DAG of $\mathcal{K}$. If $\mathcal{K}$ consists only of level 0 keys then the result follows from theorem 3.10. Suppose the maximum level in the DAG of $\mathcal{K}$ is $(n+1)$ and assume that the theorem holds for maximum level n. Let $\mathcal{K}'$ be the basis $\mathcal{B}(\mathcal{K})$ of the set of keys $\mathcal{K}$. Suppose during the protocol run, an honest party decrypts a ciphertext with key s successfully which was not produced by an honest party by encryption with s.

We build a $|\mathcal{K}'|$-IND-CCA adversary $\mathcal{A}_1$ against set of keys $\mathcal{K}'$ along the lines of the proof of theorem 3.15. In the course of interaction with $\mathcal{A}$, if $\mathcal{S}$ succeeds in decrypting a ciphertext with

key s_0 which was not produced at a previous stage by $\mathcal{S}$ by encryption with s_0, $\mathcal{A}_1$ outputs $d = 0$. Otherwise, it outputs $d = 1$. The advantage of $\mathcal{A}_1$ against the $|\mathcal{K}'|$-IND-CCA challenger is:

$$\mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{9}$$

Now, $Pr[d = 0|b = 0]$ is the probability of $\mathcal{A}_1$ succeeding in producing a ciphertext with key s which was not obtained through encryption by $\mathcal{A}_1$. $Pr[d = 0|b = 1]$ is the probability of $\mathcal{A}_1$ succeeding in decrypting a ciphertext with level-$(n-1)$ key s_0 (Same argument as in proof of theorem 3.15 - the DAG reduces by one level) which was not produced by encryption with s_0. Therefore,

$$Pr[d = 0|b = 0] = \mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) + Pr[d = 0|b = 1]$$

As the encryption scheme is IND-CCA secure, the first probability on the RHS must be negligible. The second probability is negligible due to the induction hypothesis as the encryption scheme is both IND-CCA and INT-CTXT secure. Hence the theorem. □

4 Diffie-Hellman

In this section, we formulate a trace property for protocols that use the Diffie-Hellman primitive and prove that, under the Decisional Diffie-Hellman assumption, any protocol that satisfies this condition produces keys that are suitable for keying chosen plaintext (IND-CPA) secure encryption schemes. The motivating application for this result is the fact that many Diffie-Hellman-based key exchange protocols (e.g., IKEv2 [6]) set up keys for use in secure sessions protocols. Such protocols typically provide the desired security with IND-CPA encryption schemes and do not require IND-CCA secure encryption.

Definition 4.1 (DHSafe Trace) *Let* x *and* y *be the symbolic representations of two specific nonces in a trace* $\langle e, \lambda \rangle$*. We say that* $\langle e, \lambda \rangle$ *is an* (x, y)-DHSafe trace *if the following properties hold for every thread belonging to honest principals:*

- *a thread which generates a new nonce* u *in* e*, with* $\lambda(\mathsf{u}) = \lambda(\mathsf{x})$*, ensures that it appears only exponentiated as* g^{u} *in any message sent out. Similarly for* y*.*
- *the thread generating* u*, with* $\lambda(\mathsf{u}) = \lambda(\mathsf{x})$ *is allowed to generate a key by exponentiating a term* m *such that* $\lambda(\mathsf{m}) = g^{\lambda(\mathsf{y})}$ *to the power* u *and using an appropriate key generation algorithm. However, this key is only used as a key. Similar restriction applies to* y*.*
- *the results of decryptions with the above key are not used to construct any message sent out.*

Definition 4.2 (DHSafe Protocol) *Let* x *and* y *be the symbolic representations of two specific nonces in a trace* $\langle e, \lambda \rangle$*. A protocol* $\mathcal{Q}$ *is an* (x, y)-DHSafe protocol *if for all probabilistic polytime adversaries* $\mathcal{A}$ *and for all sufficiently large security parameters* η*, the probability that a trace* $t(\mathcal{A}, \mathcal{Q}, \eta)$*, generated by the interaction of* $\mathcal{A}$ *with principals following roles of* $\mathcal{Q}$*, is a DHSafe trace with respect to* x *and* y *is overwhelmingly close to 1, the probability being taken over all adversary and protocol randomness. Formally,*

$$\forall \textit{ PPT adversary } \mathcal{A}. \exists \textit{ negligible function } \nu. \exists \eta_0. \forall \eta \geq \eta_0.$$
$$Pr[t(\mathcal{A}, \mathcal{Q}, \eta) \textit{ is DHSafe wrt } x \textit{ and } y\ ] \geq 1 - \nu(\eta)$$

As in the case of secretive protocols, here also we implicitly look at the subset of all traces that are DHSafe among all possible traces with similar justification.

Theorem 4.3 (DH-CPA security) *Assume that a probabilistic poly-time adversary interacts with an* (x,y)*-DHSafe protocol. Suppose the encryption scheme used by the protocol is IND-CPA secure and the DDH assumption holds for the group containing* g*. Then the adversary has negligible advantage at winning an IND-CPA game against a symmetric encryption challenger, using the key* $\mathsf{k} = keygen(g^{\mathsf{xy}})$*, after the interaction. In other words, a DHSafe protocol satisfies IND-CPA key usability for* k*.*

Proof. Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with an (x,y)-DHSafe protocol. We will show that if $\mathcal{A}$ has non-negligible advantage at winning an IND-CPA game against a symmetric encryption challenger, using the key k, after the interaction then we can construct either a DDH adversary $\mathcal{A}_1$ with non-negligible advantage against DDH in the group containing g or an IND-CPA adversary $\mathcal{A}_2$ with non-negligible advantage against the encryption scheme.

Adversary $\mathcal{A}_1$ is provided, at the outset, with a triple (g^a, g^b, g^c) and has to determine if $c = ab$. It proceeds by simulating the execution of the protocol to adversary $\mathcal{A}$. Following the definition of DHSafe protocols, if an honest principal sends out a message containing x or y, then it has to be constructed from g^{x} or g^{y}. $\mathcal{A}_1$ uses g^a and g^b as the bitstring representations of g^{x} and g^{y} respectively. When an honest principal exponentiates a term to the power x or y and generates a key, $\mathcal{A}_1$ uses $k = keygen(g^c)$ as the bitstring representation of the key.

In the second phase, $\mathcal{A}_1$ uniformly randomly chooses a bit b' and provides oracle $\mathcal{E}_k(LoR(\cdot,\cdot,b'))$ to $\mathcal{A}$ for an IND-CPA game. $\mathcal{A}$ finishes by outputting a bit d'. If $b' = d'$, $\mathcal{A}_1$ outputs *yes* else outputs *no*. The advantage of $\mathcal{A}_1$ against the DDH challenger is:

$$\mathbf{Adv}_{DDH,\mathcal{A}_1}(\eta) = Pr[yes|c = ab] - Pr[yes|c \neq ab] \tag{10}$$

Observe that if $c = ab$ then k is the bitstring representation of $keygen(g^{\mathsf{xy}})$. Hence, the first probability is precisely the probability of $\mathcal{A}$ winning an IND-CPA challenge with k as the key after interacting with an (x,y)-*DHSafe protocol*. We will now bound the second probability.

We start by constructing a second adversary $\mathcal{A}_2$ which has access to an encryption oracle $\mathcal{E}_k(LoR(\cdot,\cdot,b_1))$ with k unknown. It has a similar behaviour towards $\mathcal{A}$ as $\mathcal{A}_1$ had except when required to encrypt using the generated key, it queries $\mathcal{E}_k(LoR(\cdot,\cdot,b_1))$. Decryption queries are not required as results of decryptions are not used to construct any message sent. In the second phase, $\mathcal{A}_1$ uses the $\mathcal{E}_k(LoR(\cdot,\cdot,b_1))$ to provide the IND-CPA challenger to $\mathcal{A}$. $\mathcal{A}$ finishes by outputting a bit d_1 which is what $\mathcal{A}_2$ also outputs. We observe here that if $c \neq ab$ for the earlier LoR oracle, it makes no difference to the algorithm $\mathcal{A}$ whether it is interacting with $\mathcal{A}_1$ or $\mathcal{A}_2$. Thus we have:

$$(1/2)\mathbf{Adv}_{IND-CPA,\mathcal{A}_2}(\eta) = Pr[d_1 = b_1] - 1/2 = Pr[yes|c \neq ab] - 1/2 \tag{11}$$

By the equations 10 and 11 we have:

$$Pr[yes|c = ab] - 1/2 = \mathbf{Adv}_{DDH,\mathcal{A}_1}(\eta) + (1/2)\mathbf{Adv}_{IND-CPA,\mathcal{A}_2}(\eta)$$

As the probablity in the LHS is non-negligible, at least one of the advantages in the RHS must be non-negligible and hence we are done. □

5 Conclusion

We develop foundations for inductive proofs of computational security properties by proving connections between selected trace properties and useful non-trace properties. We prove that all secretive protocols have computational secrecy and authentication properties, assuming the encryption scheme used provides chosen ciphertext security and ciphertext integrity. In addition, we prove a similar theorem for computational secrecy assuming Decisional Diffie-Hellman and a chosen plaintext secure encryption scheme.

While several approaches are possible, we do not present methods for proving that a protocol is secretive. In forthcoming work, we develop a form of secrecy induction general enough to cover Kerberos and a DH induction general enough to address properties of IKEv2. Protocol proofs based on the connection between trace properties and computational secrecy developed in this paper use direct reasoning about the computational model, and do not require the stronger cryptographic assumptions that are inherent in methods based on equivalence between symbolic and computational models.

References

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

[2] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology - EUROCRYPT 2000, Proceedings*, pages 259–274, 2000.

[3] 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.

[4] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.

[5] 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.

[6] E. C. Kaufman. Internet Key Exchange (IKEv2) Protocol, 2005. RFC.

[7] 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.

[8] 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.

[9] 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.

[10] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.

[11] 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.

[12] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pages 321–334. IEEE, 2006.

[13] 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.

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

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

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

[17] 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.

[18] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *FSE*, pages 284–299, 2000.

[19] 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.

[20] 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.

[21] D. Phan and D. Pointcheval. Une comparaison entre deux methodes de preuve de securite. In *Proc. of RIVF*, pages 105–110, 2003.

[22] S. Schneider. Verifying authentication protocols with csp. *IEEE Transactions on Software Engineering*, pages 741–58, 1998.

[23] F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.