Inductive Trace Properties for Computational Security

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

Department of Computer Science, Stanford University

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 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. This 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 [7] 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 [7, 16, 11, 20]. 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 methods 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" [21] and related work in the strand space approach [22], 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., [18, 7, 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 [12, 8, 9] 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, although the attacker may produce and send arbitrary bitstrings, 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 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 [10] 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 [11]. We describe some problems with inductive reasoning about key indistinguishability and discuss the alternative condition that appears more suitable for our purposes.

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 [13]; see [19] 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 [15] 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 [11] that is parameterized by the security goal of the application in which the resulting key is used.

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 [11] 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 [14]. 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 $\langle e, \lambda \rangle$. Define $\Lambda(\mathcal{K}) = \{\lambda(k) | k \in \mathcal{K}\}$.

Definition 1 (Secretive Trace). *A trace $\langle e, \lambda \rangle$ is a* secretive trace *with respect to s and $\mathcal{K}$ if the following properties hold for every thread belonging to honest principals:*

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

Definition 2 (Secretive Protocol). *A protocol $\mathcal{Q}$ is a* secretive protocol *with respect to s and $\mathcal{K}$ 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 a secretive trace with respect to s and $\mathcal{K}$ is overwhelmingly close to 1, the probability being taken over all adversary and protocol randomness. Formally,*

$$\forall \; PPT \; adversary \; \mathcal{A}. \; \exists \; negligible \; function \; \nu. \; \exists \eta_0. \; \forall \eta \geq \eta_0.$$
$$Pr[t(\mathcal{A}, \mathcal{Q}, \eta) \; is \; secretive \; wrt \; s \; and \; \mathcal{K} \;] \geq 1 - \nu(\eta)$$

A *level-0* key for a protocol execution is an encryption key which is only used as a key but never as 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.

Theorem 1 (CCA security - No keying - level 1). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of level-0 keys $\mathcal{K}$. 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. Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with a secretive protocol with respect to nonce s and a set of level-0 keys $\mathcal{K}$. We will show that if $\mathcal{A}$ has non-negligible advantage at distinguishing 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 $\mathcal{A}_1$ has access to multi-party Left-or-Right encryption oracles $\mathcal{E}_{k_i}(LoR(\cdot, \cdot, b))$ parameterized by a 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). $\mathcal{A}_1$ will provide a simulation of the *secretive protocol* to $\mathcal{A}$ by using these oracles. $\mathcal{A}_1$ randomly chooses two nonces x_0, x_1 at the outset as alternate bit-string representations of s. Suppose $u(s, \cdots)$ is a term explicitly constructed from s. As $\mathcal{A}_1$ 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 $\mathcal{A}$. In this case $\mathcal{A}_1$ asks the encryption oracle $(u(x_0, \cdots), u(x_1, \cdots))$ to be encrypted by k. If a message construction involves decryption with a key in $\mathcal{K}$, $\mathcal{A}_1$ first checks whether the term to be decrypted was produced by an encryption oracle - if not then the decryption oracle is invoked; if yes then $\mathcal{A}_1$ uses the corresponding encryption query as the decryption. In the second case the encryption query might have been of the form $(v(x_0, \cdots), v(x_1, \cdots))$. 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.

In the second phase, $\mathcal{A}_1$ chooses a bit d' and sends $x_{d'}$ 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 2 (CCA security - Keying - level 1). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of level-0 keys $\mathcal{K}$. 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 a secretive protocol with respect to nonce s and a set of level-0 keys $\mathcal{K}$. 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 1 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 $\mathcal{A}_1$ encrypts or decrypts with x_0.

In the second phase, $\mathcal{A}_1$ uniformly randomly chooses a bit b' and provides oracles $\mathcal{E}_{x_0}(LoR(\cdot, \cdot, b'))$ and $\mathcal{D}_{x_0}(\cdot)$ 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 x_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 a *secretive protocol* w.r.t. s and $\mathcal{K}$. We will now bound the second probability. We start by constructing a second adversary $\mathcal{A}_2$ which has all the keys in $\mathcal{K}$, randomly generates a nonce x_1 and has access to an encryption oracle $\mathcal{E}_{x_0}(LoR(\cdot, \cdot, b_1))$ and a decryption oracle $\mathcal{D}_{x_0}(\cdot)$. It has a similar behaviour towards $\mathcal{A}$ as $\mathcal{A}_1$ had except that when constructing terms with s, it uses x_1 but when required to encrypt or decrypt using s, it queries $\mathcal{E}_{x_0}(LoR(\cdot, \cdot, b_1))$ or $\mathcal{D}_{x_0}(\cdot)$. In the second phase, $\mathcal{A}_1$ uses the oracles $\mathcal{E}_{x_0}(LoR(\cdot, \cdot, b_1))$ and $\mathcal{D}_{x_0}(\cdot)$ 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:

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

By the equations 2 and 3 we have:

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

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 [17].

Theorem 3 (CTXT integrity - level 1). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of level-0 keys $\mathcal{K}$. 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.*

The proof is outlined in Appendix A. We now extend theorems 1–3 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 (Key Graph). *Let $\mathcal{K}$ be the symbolic representations of nonces and keys associated with a specific thread 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 4 (Key Level). *Consider a directed acyclic key graph. 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 5 (Key Closure). *For a set of keys $\mathcal{K}$ from a directed acyclic key graph, we define its closure $\mathcal{C}(\mathcal{K})$ to be the union of sets of keys at the root which are predecessors of each key in $\mathcal{K}$.*

Theorem 4 (CCA security - No Keying). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of keys $\mathcal{K}$ in a DAG of finite and statically bounded level. 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 1. 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 closure $\mathcal{C}(\mathcal{K})$ of the set of keys $\mathcal{K}$.

Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with a secretive protocol with respect to nonce s and the set of keys $\mathcal{K}$. 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 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, $\mathcal{A}_1$ chooses random values x_0^i, x_1^i and for s, $\mathcal{A}_1$ chooses random values x_0, x_1. $\mathcal{A}_1$ constructs messages to be sent to $\mathcal{A}$ as follows:

- to encrypt the term $f(s, s^1, s^2, ...)$ with $k_i \in \mathcal{K}'$, use response to oracle query $\mathcal{E}_{k_i}(f(x_0, x_0^1, x_0^2, ...), f(x_1, x_1^1, x_1^2, ...), b)$.
- to encrypt $f(s, s^1, s^2, ...)$ with s^i, use $\mathcal{E}_{x_0^i}(f(x_0, x_0^1, x_0^2, ...))$.

Decryption operations are served analogously.

In the second phase, $\mathcal{A}_1$ chooses a bit d' and sends $x_{d'}$ 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:

$$\begin{aligned} \mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) &= Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \\ &= (Pr[d = 0|b = 0] - 1/2) + (Pr[d = 1|b = 1] - 1/2) \quad (4) \end{aligned}$$

The first probability in the RHS is precisely the probability of $\mathcal{A}$ breaking the indistinguishability of x_0 or equivalently of s. In the case when $b = 1$, the terms were constructed in the following manner:

- encrypt $f(s, s^1, s^2, ...)$ with $k_i \in \mathcal{K}'$: $\mathcal{E}_{k_i}(f(x_1, x_1^1, x_1^2, ...))$.
- encrypt $f(s, s^1, s^2, ...)$ with s^i: $\mathcal{E}_{x_0^i}(f(x_0, x_0^1, x_0^2, ...))$.

We observe here that $\mathcal{A}_1$ simulated the execution of another secretive protocol $\mathcal{G}'$ with keys of level $\leq n$ - $x_0^1, x_0^2, ...$ protecting x_0. 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:

$$Pr[d = 1|b = 1] - 1/2 = (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta) \quad (5)$$

By the equations 4 and 5 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 5 (CCA security - Keying). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of keys $\mathcal{K}$ in a DAG of finite and statically bounded level. 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 2. 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 closure $\mathcal{C}(\mathcal{K})$ of the set of keys $\mathcal{K}$.

Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with a secretive protocol with respect to nonce s and the set of keys $\mathcal{K}$. 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 4 to construct the adversary $\mathcal{A}_1$. The only additional operation is that to encrypt or decrypt the term m with s, we use x_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 $\mathcal{E}_{x_0}(m_{b'})$. Decryption requests are also served by decrypting with key x_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:

$$\mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d=0|b=0] - Pr[d=0|b=1] \tag{6}$$

The first probability is precisely the probability of $\mathcal{A}$ breaking the 'good-key'-ness of x_0 or equivalently of s. In the case when $b = 1$, the terms were constructed in the following manner:

- encrypt $f(s, s^1, s^2, ...)$ with $k_i \in \mathcal{K}'$: $\mathcal{E}_{k_i}(f(x_1, x_1^1, x_1^2, ...))$.
- encrypt $f(s, s^1, s^2, ...)$ with s^i: $\mathcal{E}_{x_0^i}(f(x_0, x_0^1, x_0^2, ...))$.
- encrypt term m with s: $\mathcal{E}_{x_0}(m)$.

We observe here that $\mathcal{A}_1$ simulated the execution of another secretive protocol $\mathcal{G}'$ with keys of level $\leq n$ - $x_0^1, x_0^2, ...$ protecting x_0. 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:

$$Pr[d=0|b=1] - 1/2 = (1/2)\mathbf{Adv}_{\mathcal{G}',\mathcal{A}}(\eta) \tag{7}$$

By the equations 6 and 7 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 6 (CTXT integrity). *Assume that a probabilistic poly-time adversary interacts with a secretive protocol with respect to nonce s and a set of keys $\mathcal{K}$ in a DAG of finite, statically bounded levels. 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.*

The proof is outlined in Appendix A.

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

- *a thread which generates a new nonce u in e, with $\lambda(u) = \lambda(x)$, ensures that it appears only exponentiated as g^u in any message sent out. Similarly for y.*
- *the thread generating u, with $\lambda(u) = \lambda(x)$ is allowed, at most once, to generate a key by exponentiating any term m of the appropriate type 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 7 (DHSafe Protocol). *Let x and y be the symbolic representations of two nonces associated with a specific thread in a trace $\langle e, \lambda\rangle$. A protocol $\mathcal{Q}$ is a* DHSafe protocol *with respect to x and y 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 a DHSafe trace with respect to s and $\mathcal{K}$ is overwhelmingly close to 1, the probability being taken over all adversary and protocol randomness. Formally,*

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

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 7 (DH-CPA security). *Assume that a probabilistic poly-time adversary interacts with a DHSafe protocol with respect to nonces x and y. The adversary has negligible advantage at winning an IND-CPA game against a symmetric encryption challenger, using the key $k = keygen(g^{xy})$, after the interaction if the encryption scheme is IND-CPA secure and the DDH assumption holds for the group containing g. 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 a DHSafe protocol with respect to nonces x and y. 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 $k = \mathit{keygen}(g^c)$ as the bitstring representation of the key.

In the second phase, $\mathcal{A}_1$ uniformly randomly chooses a bit b' and provides oracle $\mathcal{E}_k(\mathit{LoR}(\cdot,\cdot,b'))$ 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:

$$\mathbf{Adv}_{DDH,\mathcal{A}_1}(\eta) = Pr[yes|c = ab] - Pr[yes|c \neq ab] \tag{8}$$

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

We start by constructing a second adversary $\mathcal{A}_2$ which has access to an encryption oracle $\mathcal{E}_k(\mathit{LoR}(\cdot,\cdot,b_1))$ 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 $\mathcal{E}_k(\mathit{LoR}(\cdot,\cdot,b_1))$. 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 $\mathcal{E}_k(\mathit{LoR}(\cdot,\cdot,b_1))$ 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:

$$(1/2)\mathbf{Adv}_{IND-CPA,\mathcal{A}_2}(\eta) = Pr[d_1 = b_1] - 1/2 = Pr[yes|c \neq ab] - 1/2 \tag{9}$$

By the equations 8 and 9 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. $\square$

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 methods 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. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. A. Freier, P. Karlton, and P. Kocher. The SSL protocol version 3.0. IETF Internet draft, November 18 1996.
14. O. Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge University Press, 2004.
15. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.
16. 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.
17. J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *FSE*, pages 284–299, 2000.
18. 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.

19. 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.
20. D. Phan and D. Pointcheval. Une comparaison entre deux methodes de preuve de securite. In *Proc. of RIVF*, pages 105–110, 2003.
21. S. Schneider. Verifying authentication protocols with csp. *IEEE Transactions on Software Engineering*, pages 741–58, 1998.
22. F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

A Additional Proofs

Proof of theorem 3. Assume that a probabilistic poly-time adversary $\mathcal{A}$ interacts with a secretive protocol with respect to nonce s and a set of level-0 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 $\mathcal{K}$ in the lines of the proof of theorem 2. However, this new $\mathcal{A}_1$ computes d in a different way. Recall that $\mathcal{A}_1$ uses x_0 when it intends to encrypt or decrypt using s. In the course of interaction with $\mathcal{A}$, if $\mathcal{A}_1$ succeeds in decrypting a ciphertext with key x_0 which was not produced at a previous stage by $\mathcal{A}_1$ by encryption with x_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:

$$\mathbf{Adv}_{|\mathcal{K}|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{10}$$

Now, $Pr[d = 0|b = 0]$ is the probability of $\mathcal{A}_1$ succeeding in decrypting 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-0 key x_0 (as in this case x_0 was only used as a key). Therefore, using a similar idea as proof of theorem 2 we can build an INT-CTXT adversary $\mathcal{A}_2$ against x_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.

□

Proof of theorem 6. 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. 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 closure $\mathcal{C}(\mathcal{K})$ 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 5. In the course of interaction with $\mathcal{A}$, if $\mathcal{A}_1$ succeeds in decrypting a ciphertext with key x_0 which was not produced at a previous stage by $\mathcal{A}_1$ by encryption with x_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:

$$\mathbf{Adv}_{|\mathcal{K}'|-IND-CCA,\mathcal{A}_1}(\eta) = Pr[d = 0|b = 0] - Pr[d = 0|b = 1] \tag{11}$$

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 x_0 (Same argument as in proof of theorem 5 - the DAG reduces by one level) which was not produced by encryption with x_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.

□