Computational Soundness of Formal Reasoning about Indistinguishability and Non-Malleability of Cryptographic Expressions

by

Mohammad Hajiabadi
B.Sc., Sharif University of Technology, 2009

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Computational Soundness of Formal Reasoning about Indistinguishability and Non-Malleability of Cryptographic Expressions

by

Mohammad Hajiabadi
B.Sc., Sharif University of Technology, 2009

Supervisory Committee

Dr. Bruce Kapron, Supervisor
(Department of Computer Science)

Dr. Venkatesh Srinivasan, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Bruce Kapron, Supervisor
(Department of Computer Science)

Dr. Venkatesh Srinivasan, Departmental Member
(Department of Computer Science)

ABSTRACT

Analysis and verification of security protocols are typically carried out in two different models of cryptography: *formal cryptography* and *computational cryptography*. Formal cryptography, originally inspired by the work of Dolev and Yao [14], takes an abstract and idealized view of security, and develops its proof techniques based on methods and ideas from logic and theory of programming languages. It makes strong assumptions about cryptographic operations by treating them as perfectly-secure symbolic operations. Computational cryptography, on the other hand, has developed its foundations based on complexity theory. Messages are viewed as bit-strings, and cryptographic operations are treated as actual transformations on bit-strings with certain asymptotic properties.

In this thesis, we explore the relation between the Dolev-Yao model and the computational model of public-key cryptography in two contexts: *indistinguishability* and *non-malleability of expressions*. This problem in the absence of *key-cycles* is partially addressed in [21, 20] by Herzog. We adapt our approach to use the *co-inductive* definition of symbolic security, whose private-key treatment was considered in [27], and establish our main results as follow:

- Using a co-inductive approach, we extend the indistinguishability and non-malleability results of Herzog in the presence of key-cycles.
- By providing a counter-example, we show that the indistinguishability property in this setting is strictly stronger than the non-malleability property, which gives a negative answer to Herzog's conjecture that they are equivalent.

- we prove that despite the fact that IND-CCA2 security provides non-malleability in our setting, the same result does not hold for IND-CCA1 security.
- We prove that, under certain hypothesis, our co-inductive formal indistinguishability is computationally-complete in the absence of key-cycles and with respect to any *length-revealing* encryption scheme. In the presence of key-cycles, we prove that the completeness does not hold even with respect to IND-CPA security.

Contents

ACKNOWLEDGEMENTS

My thanks go first and foremost to my supervisor, Dr. Bruce Kapron. He introduced me to the wonderful world of cryptography, and the helpful discussions we had led to the development of ideas in this thesis. Without his encouragement and guidance, this thesis would have never been written. I would also like to thank my thesis committee members, Dr. Venkatesh Srinivasan and Dr. Audrey Yap. I am especially grateful to Dr. Srinivasan for his invaluable advice when I was dealing with my PhD applications.

I owe my deepest gratitude to my very dear friends, Kazem and Khalegh . They have always been extremely supportive to me in times of difficulty, and taught me very good lessons that I will never forget.

Last but definitely not least, I wish to express my affectionate thanks to my family, especially to my mother: "I am where I am because of you". Love you.

DEDICATION

To my parents, for their unconditional love and
never-ending support.

Chapter 1

Introduction

1.0.1 Overview of Formal and Computational Views

Verifying the correctness of security protocols is a fundamental and challenging task in cryptography. The rigorous analysis of security protocols typically follows one of two approaches, based either on formal cryptography, or on computational cryptography. In formal cryptography (e.g., [14, 1, 12]), messages are described as formal expressions built upon some term algebra, and cryptographic primitives are considered as purely syntactic operations on them. Proofs of correctness for security protocols in formal cryptography are usually based on the assumption that the underlying cryptographic primitives participating in security protocols are ideally secure. As an example, for the encryption primitive, it is assumed that the only way for a formal adversary to obtain any information from a given ciphertext is to have the underlying key. Because of its high level of abstraction, it offers convenient methods (e.g. automated tools) for reasoning about cryptographic protocols. However, proofs of security established in this idealized model are not directly transferable to the more realistic, computational model, where cryptographic operations are treated as *efficient* (i.e. *probabilistic polynomial-time*) algorithms satisfying certain asymptotic properties. Security proofs in the computational model are carried out via *reduction*: an encryption system is secure if its security is implied by some problem known to be computationally hard.

1.0.2 Motivation

In the last few years, starting with the seminal work of Abadi and Rogaway [3], there has been a lot of effort in relating these two views of cryptography. Abadi and Rog-

away [3] develops a logic for reasoning about indistinguishability of expressions with two semantics: formal semantics and computational semantics. Expressions in this logic are built over some atomic messages, which include a set of key symbols and basic terms, using two symbolic operations for creating compound messages: *encryption* and *concatenation*. For example, an expressions like $(\{m\}_k, k_1)$ may denote the concatenation of two messages where the first one, in turn, denotes the encryption of message m under key k and the second one denotes a key symbol k_1. Each semantics gives rise to a notion of *equivalence* (i.e. *indistinguishability*) between expressions. The notion of equivalence in the computational semantics is that of *standard notion of computational indistinguishability*, where it relates two expressions if their naturally-associated probability distributions are computationally-indistinguishable. In the formal semantics, the notion of equivalence is defined by associating a *pattern* with each formal expression, where the pattern of an expression is obtained by replacing each undecipherable part of the expression by a special symbol $\Box$ which intuitively symbolizes an undecryptable message. Now two expressions are formally-equivalent if they yield the same pattern. Abadi and Rogaway prove that, under sufficiently strong security conditions, formal reasoning about indistinguishability of expressions is computationally-sound. That is, if two expressions are proven to be equivalent with respect to the formal semantics, their computational interpretations are computationally-indistinguishable.

However, the result of Abadi and Rogaway requires the expressions to be *encryption-cycle* (*key-cycle*) free. The simplest form of an encryption cycle happens when a key encrypts itself. Encryption schemes satisfying the standard notion of *semantic security* of [19] (and even its stronger notion of security against *chosen-ciphertext attack* [31]) do not necessarily remain secure in the presence of key-cycles (i.e. if the adversary is allowed to have encryptions of messages which may depend on the underlying secret key). In fact, from any semantically-secure encryption scheme, one can construct another semantically-secure encryption scheme which becomes totally insecure if the adversary is given the encryption of the underlying secret-key under its public-key (in the setting of private-key encryption, both these keys are the same). The issue of key-cycles demonstrates some kind of discrepancy between the formal and computational treatments of encryption. Although the key-cycle usage in the computational model is proven to be dangerous, the formal model simply postulates that the occurrence of key-cycles poses no security threat. As an example, in formal cryptography, encrypting a key under itself is considered to be secure in the sense that no

formal adversary can distinguish it from any other encryption as long as it does not have the underlying key symbol.

In the literature, two main approaches have been taken in order to resolve the mentioned discrepancy between the two models: strengthening the computational model [10, 6], or weakening the formal model [24, 27]. In [10], a very strong notion of computational security, called *KDM-security*, which stands for *key-dependent messages*, is developed, with respect to which the adversary may request to receive encryptions of plaintexts of its own choice, possibly depending on the underlying secret key. Adao et al. in [6] prove that the Abadi-Rogaway formal encryption is sound with respect to the computational semantics satisfying the KDM-security condition (see the "previous work" sub-chapter for further exposition on this matter). The other approach [27], which is also the approach taken in our work, attempts to refine the definition of symbolic security for key-cyclic expressions by employing a *co-inductive* definition for formulating the adversarial knowledge set, as opposed to *inductive* definitions considered in previous work. The result of [27] indicates that in the presence of key-cycles, if two expressions are symbolically equivalent, their computational interpretations (via a CPA-secure symmetric encryption schemes) are computationally indistinguishable.

Apart from indistinguishability, the relation between formal and computational models of cryptography has been considered in the context of non-malleability [21, 20] as well. In this context, the goal of the adversary is not to distinguish between two expressions, but to transform one of them to the other. The formal adversary is confined to certain fixed operations to perform this transformation; that is, from a given message e, it can produce messages in the *closure* of e, a set which is defined based on Dolev-Yao deduction rules. We say an encryption scheme provides the non-malleability property if no adversary given the computational interpretation of e can produce the computational interpretation of any message outside $closure(e)$, except with negligible probability. To the best of our knowledge, the work of Herzog ([21, 20]) is the only place where the relation between the two views is explored in terms of non-malleability. However, there are still some unexplored problems regarding the relation between the non-malleability and the indistinguishability property which deserve attention, and we try to address some of them in this work.

1.0.3 Our Results

In this thesis, we focus on formal and computational models based on public-key cryptography, and we extend the results of [21] in several directions. In [21], a stronger version of computational indistinguishability is developed, in which the distinguisher is granted a decryption oracle, and it is proved that in the absence of key-cycles, if an encryption scheme provides IND-CCA2 security, it also provides this strong indistinguishability property. Moreover, it is proved that in the absence of key-cycles, if an encryption scheme provides the indistinguishability property, it also provides the non-malleability property. First of all, the counter example of [6] which shows that IND-CCA2 security does not provides security under circular-encryption already implies that, in the inductive setting, IND-CCA2 security is not sufficient for providing non-malleability in the presence of key-cycles. In our work, we try to resolve this issue in the co-inductive setting. In particular, we consider the co-inductive definition of symbolic security, as in [27], but in the setting of public-key encryption, and we re-define the notions of strong indistinguishability and non-malleability in our framework, which we call *co-inductive strong indistinguishability* and *co-inductive non-malleability*, respectively. Specifically, our contributions include:

- We show that in the presence of key cycles, IND-CCA2 secrecy provides co-inductive strong indistinguishability, extending the result of [21]. That is, we show that if two formal expressions are co-inductively equivalent, their computational interpretations (via IND-CCA2 secure encryption schemes) are strongly indistinguishable (Corollary 1). The proof of this fact is much more difficult than the soundness result of [27] because of the distinguisher's access to decryption oracles.

- We show that the result of [21] which states that indistinguishability implies non-malleability, extends to the co-inductive framework. As an implication, we show that, in the presence of key-cycles, IND-CCA2 secure encryption schemes provide co-inductive non-malleability (Theorem 4).

- By giving a counter-example, we show that, in both the inductive and co-inductive settings, indistinguishability is strictly stronger than non-malleability, which provides a negative answer to Herzog's conjecture that they are equivalent. (Theorem 5)

- By providing a counter-example, we show that in both the inductive and co-inductive settings, if we weaken the security condition from IND-CCA2 to IND-CCA1, the non-malleability property is no longer satisfied (Theorem 6) .
- We show that in the presence of key-cycles, *IND-CPA* security does not give a computationally-complete interpretation (Theorem 7), and we prove that in the absence of key-cycles, the completeness result holds not only with respect to IND-CPA secure encryption systems, but also with respect to any length-revealing (not necessarily secure) encryption scheme (Claim 1).

1.0.4 Previous Work

Computational soundness of equivalence for formal expressions has been addressed by many papers in recent years (e.g. [3, 24, 2, 5, 7, 4, 6, 21, 27, 26, 4]. The computational completeness problem is also studied in [29, 22]. In particular, the result of [29] demonstrates that type-0 security, with respect to which it was proved that formal equivalence is computationally sound, does not provide completeness; completeness is obtained by strengthening the computational security condition to satisfy a stronger notion of security called *confusion-freeness*, which, informally speaking, states that decryption with wrong keys fails. In both [29, 22], the computational completeness problem is considered in the setting of private-key encryption. In our work, however, we demonstrate that in the setting of public-key encryption and under reasonable assumptions, the completeness result holds in the absence of key-cycles with respect to any length-revealing encryption scheme (not necessarily secure).

Computational soundness of formal equivalence in the presence of key-cycles has been addressed in a number of papers (e.g. [24, 27, 6, 25]). In [24], computational soundness in the presence of key cycles is obtained by giving more deductive power to the formal adversary. In particular, Laud modifies the set of deduction rules of Abadi-Rogaway logic by adding a specific rule which enables the formal adversary to break a key-cyclic expression. In [6], it is proved that even security against chosen-ciphertext attack (as one of the strongest notions of security in the standard model) does not guarantee computational soundness in the presence of key-cycles, but computational soundness may be obtained using a very strong notion of security, called KDM security [10]. KDM security is a very strong notion of security which, informally speaking, has the property that it remains secure even if the adversary is provided with encryptions of plaintexts (of her choice) which may depend on the underlying secret key. No

construction of an encryption system was known in the standard model to provably meet this notion of security, until the quite recent work of Boneh et al. [11] which constructs such an encryption system under the Decision Deffie-Hellman assumptions.

The Abadi-Rogaway logic was later extended to include other primitives than encryption. Garcia and Rossum [16] enrich the Abadi-Rogaway logic by including an operator for *formal hashes*, and they prove that if formal hashes are interpreted as perfectly one-way functions in the computational world, then the type-0 security condition, with respect to which the Abadi-Rogaway logic was proved to be sound, also provides soundness in this generalized setting. Micciancio and Panjwani [28] strengthen the Abadi-Rogaway's adversarial model by considering more adaptive adversaries, where the adversary can get to see the computational interpretations of a sequence of adaptively-chosen expressions. In contrast, the Abadi-Rogaway's framework models passive adversaries with very limited power, which can eavesdrop on a communication line between two parties, and can just see the messages exchanged on this line. In order to formulate their computational soundness problem, Micciancio and Panjwani consider the adversary operating in two worlds: in the first world, the adversary receives the computational evaluations of its (adaptively) chosen expressions, and in the second world, it receives the computational evaluations of their patterns. Now a computational encryption provides computational soundness in this framework if, when used for computational evaluation, the adversary cannot determine with which world it was interacting with a probability non-negligibly greater than $\frac{1}{2}$. They prove that under reasonable syntactic restrictions on the adversary's chosen expressions, most of which common to previous work, the computational soundness result holds with respect to IND-CPA security (see also [30] for a treatment of active adversaries).

In all these works, adversarial knowledge in the formal setting is formulated using an inductive approach. In [27], Micciancio suggests a co-inductive method for formulating adversarial knowledge, and proves that, in such a setting, the Abadi-Rogaway's soundness property extends in the presence of key-cycles. In [26], using the co-inductive approach, he extends his previous computational soundness result for expressions with pseudo-random keys in the presence of key-cycles.

Chapter 2

Formal Encryption

2.0.5 Language of Expressions

In this chapter, we review the generalization of Abadi-Rogaway logic to the case of asymmetric encryption, given in [21]. Let K_{pub} and K_{priv} be a set of public and private keys, and $Block$ be a fixed set (disjoint from K_{pub} and K_{priv}) containing some basic messages. Compound messages are constructed by the application of two syntactic operations: the *pairing operation* and the *encryption operation*. More formally, the set of *formal expressions* is given by the following grammar:

$$Exp ::= Block \mid K_{priv} \mid K_{pub} \mid \{Exp\}_{K_{pub}} \mid (Exp, Exp)$$

If $e \in Exp$ and $k \in K_{pub}$, $\{e\}_k$ denotes the encryption of e under k. As in [21], we assume that there exists a bijection $inv : K_{pub} \rightarrow K_{priv}$ which maps the set of public keys to their private keys. We write K^{-1} to denote $inv(K)$ if $K \in K_{pub}$, and $inv^{-1}(K)$ if $K \in K_{priv}$. Also, if T is a set of public or private keys, we define $T^{-1} = \{K^{-1} : K \in T\}$. Throughout this work, when referring to K^{-1}, it can be realized from the context whether K^{-1} denotes some public key or private key symbol. If $e_1, \ldots, e_k$ are expressions, we write $(e_1, \ldots, e_k)$ as an abbreviation for $(\ldots(((e_1, e_2), e_3), e_4)\ldots)$.

We assume that the encryption might reveal certain information about the underlying plaintext. In particular, we assume the length and the *structure* of the plaintext are deducible from the encryption. The structure of a message is defined as follow:

- if $e \in Block$, $struct(e) = \Box$
- if $e \in K_{pub}$, $struct(e) = \circ$

- if $e \in K_{priv}$, $struct(e) = \circ_p$
- if $e = (e_1, e_2)$, $struct(e) = (struct(e_1), struct(e_2))$
- if $e = \{e_1\}_K$, $struct(e) = \{struct(e_1)\}_\circ$

That is, if the formal adversary is given the ciphertext $\{e\}_k$, the only information that the adversary can obtain about the underlying plaintext e is $struct(e)$. Having defined the *struct* function, we extend the class of expressions we consider to include *patterns*, defined as follows:

$$Pat ::= Exp \mid \{struct(Exp)\}_K \mid \{Pat\}_K \mid (Pat, Pat).$$

Henceforth we refer to the elements of Exp as expressions, and to the elements of Pat as patterns. We also refer to patterns of the form $\{struct(Exp)\}_K$ as *blobs* . Motivated by the above discussion, we can define a *pattern function* P, which takes as input a set T of private keys and a message e, and outputs the pattern of e that is visible to an adversary having access to the keys in T. Formally, for $e \in Pat$ and $T \subseteq K_{priv}$, we define:

$$\begin{aligned}
&if\ b \in Block \cup K_{pub} \cup K_{priv} \cup \{struct(e) | e \in Exp\}, p(b, T) = b \\
&if\ e = (e_1, e_2),\ p(e, T) = (p(e_1, T), p(e_2, T)) \\
&if\ e = \{e_1\}_k,\ p(e, T) = \begin{cases} \{p(e_1, T)\}_K & K^{-1} \in T \\ \{struct(e_1)\}_K & \text{otherwise} \end{cases}
\end{aligned}$$

Example 1. *Suppose* $e = (\{\{1\}_{K_1}\}_{K_2}, \{0\}_{K_3})$. *We have:*

$$\begin{aligned}
e_1 &= p(e, \{K_3^{-1}\}) = \{\{\{\square\}_\circ\}_{K_2}, \{0\}_{K_3}) \\
e_2 &= p(e_1, \{K_2^{-1}\}) = \{\{\{\square\}_\circ\}_{K_2}, \{\square\}_{K_3})
\end{aligned}$$

2.0.6 Symbolic Equivalence: Induction vs. Co-Induction

Symbolic (formal) equivalence captures the idea of when two expressions look the same to a formal adversary with no prior knowledge. For example, for $b, b' \in Bool$, we consider the two formal expressions $\{b\}_k$ and $\{b'\}_k$ to be equivalent. The reason is clear; the adversary does not have the corresponding secret key, and so is not able to

decrypt any of them and distinguish between them. On the other hand, if $k_1 \in K_{pub}$, the two expressions $\{b\}_k$ and $\{k_1\}_k$ are not equivalent. Here the scenario is quite different; although the adversary is not able to obtain the required secret key from these two ciphertexts, but it can infer that the underlying plaintexts have different structures, and, thence, is able to distinguish between the two ciphertexts. Technically, two expressions look the same to an adversary if the adversary is not able to tell their difference based on its *knowledge set*. In order to model the adversarial knowledge set, we have to specify what kind of operations the formal adversary is allowed to perform during the execution of a protocol. For simplicity, we consider the case of *passive adversaries*, where the adversary can just eavesdrop on the communication line and record the exchanged messages, without the ability to alter the control flow of the protocol, modify a transmitted message or inject a new message. In particular, we assume the formal adversary is limited to performing the following operations, inspired by the work of Dolev and Yao [14]:

- Encrypting a known message e with a public key k,
- Decryption with respect to a known secret key,
- Pairing two known elements together, and
- Separation of a pair into two elements.

Based on the above deduction rules, we may associate a *key recovery function* $\mathcal{F}_e$ to ever expressions e which takes as input $T \subseteq \mathcal{K}_{priv}$ and $e \in Pat$, and returns the set of private keys that can be recovered from e by an adversary observing e and using set T for decryption. Formally $\mathcal{F}_e(T)$ is defined as follow:

$$
\begin{aligned}
&if\ K \in K_{priv}, \mathcal{F}_K(T) = K \\
&if\ b \in Block, b \in Struct,\ or\ b \in K_{pub}, \mathcal{F}_b(T) = \emptyset \\
&if\ e = (e_1, e_2), \mathcal{F}_e(T) = \mathcal{F}_{e_1}(T) \cup \mathcal{F}_{e_2}(T) \\
&if\ e = \{e_1\}_K,\ and\ k^{-1} \in T : \mathcal{F}_e(T) = \mathcal{F}_{e_1}(T) \\
&if\ e = \{e_1\}_K,\ and\ k^{-1} \notin T; \mathcal{F}_e(T) = \emptyset.
\end{aligned}
$$

Example 2. *Assume* $e = (\{\{K_1^{-1}\}_{K_2}\}_{K_3}, K_2^{-1})$. *Then*

$$\mathcal{F}_e(\{K_3^{-1}\}) = \{K_2^{-1}\}$$
$$\mathcal{F}_e(\{K_3^{-1}, K_2^{-1}\}) = \{K_1^{-1}, K_2^{-1}\}.$$

Note that $\mathcal{F}_e(T)$ intuitively represents the set of keys which can be "immediately" inferred from e, using only keys of T for decryption. For instance, in the above example, although K_1^{-1} can be recovered from e by using the key set $\{K_3^{-1}\}$, but it is not immediately recoverable, and thus $K_1^{-1} \notin F_e(\{K_3^{-1}\})$.

We define the binary *subexpression relation* $\sqsubseteq$ over patterns as follow: $\sqsubseteq$ is the least reflexive transitive relation having the following properties: $k \sqsubseteq \{m\}_k$, $m \sqsubseteq \{m\}_k$, $e_1 \sqsubseteq (e_1, e_2)$, and $e_2 \sqsubseteq (e_1, e_2)$. Using the subexpression relation, we can define two functions $pubkeys(.)$ and $privkeys(.)$ which give the set of public and private keys occurring in a pattern. Formally we define:

- $pubkeys(e) = \{k | k \sqsubseteq e \ \&\ k \in \mathcal{K}_{pub}\}$
- $privkeys(e) = \{k | k \sqsubseteq e \ \&\ k \in \mathcal{K}_{priv}\}$

As discussed earlier, with any pattern e we may associate a key recovery function $\mathcal{F}_e$. We say that a set T of private keys is a *fixed point of* $\mathcal{F}_e$ if $\mathcal{F}_e(T) = T$. A set T_1 is the *greatest* fixed point of $\mathcal{F}_e$ if for every T that $\mathcal{F}_e(T) = T$, it holds that $T \subseteq T_1$. Similarly, a set T_1 is the *least* fixed point of $\mathcal{F}_e$ if for every T that $\mathcal{F}_e(T) = T$, it holds that $T_1 \subseteq T$. In the following theorem, as its private key version in [27], we show the existence of the least fixed point and the greatest fixed point of $\mathcal{F}_e$, which we denote by $fix(\mathcal{F}_e)$ and $FIX(\mathcal{F}_e)$ respectively:

Lemma 1. *Suppose that* $e \in Pat$ *and* $\mathcal{F}_e$ *is its associated key recovery function. Letting* $n_1 = |privkeys(e)|$*, we have:*

$$fix(\mathcal{F}_e) = \mathcal{F}_e^{n_1}(\emptyset) = \bigcup_i \mathcal{F}_e^i(\emptyset)$$
$$FIX(\mathcal{F}_e) = \mathcal{F}_e^{n_1}(privkeys(e)) = \bigcap_i \mathcal{F}_e^i(privkeys(e)).$$

Proof. We prove that $fix(F_e) = F_e^{n_1}(\emptyset)$ is the least fixed point of F_e. For this purpose, we need to first prove that $F_e^{n_1}(\emptyset)$ is actually a fixed point. This is easy to see. Consider the following sequence which contains $n_1 + 1$ sets of keys :

$$F_e^0(\emptyset) = \emptyset \subseteq F_e^1(\emptyset) \subseteq \cdots \subseteq F_e^{n_1}(\emptyset)$$

Note that for all i, $F_e^i(\emptyset) \subseteq privkeys(e)$. Now given the fact that $privkeys(e)$ has n_1 different subsets, there exists some i such that $0 \leq i \leq n_1$ and $F_e^i(\emptyset) = F_e^{i+1}(\emptyset)$. This implies that $F(F_e^{n_1}(\emptyset)) = F_e^{n_1}(\emptyset)$, which shows that $F_e^{n_1}(\emptyset)$ is a fixed point.

Now we prove that if T is a fixed point, then $F_e^{n_1}(\emptyset) \subseteq T$, which implies that $F_e^{n_1}(\emptyset)$ is the least fixed point. Note that since T is a fixed point, we have: $T = F_e^{n_1}(T)$. So we need to prove that $F_e^{n_1}(\emptyset) \subseteq F_e^{n_1}(T)$. This is easy to see because F_e is a monotone function (i.e. $T_1 \subseteq T_2 \Rightarrow F_e(T_1) \subseteq F_e(T_2)$). In a similar manner, we can prove that $F_e^{n_1}(privkeys(e))$ is the greatest fixed point of F_e.

Therefore, $fix(F_e) = F_e^{n_1}(\emptyset)$ and $FIX(F_e) = F_e^{n_1}(privkeys(e))$. Since F_e is monotone and $\emptyset \subseteq F_e(\emptyset)$, we have $F_e^i(\emptyset) \subseteq F_e^{i+1}(\emptyset)$, and thus $fix(F_e) = F_e^{n_1}(\emptyset) = \bigcup_i F_e^i(\emptyset)$. Similarly, since $F_e(privkeys(e)) \subseteq privkeys(e)$, it holds that $F_e^{i+1}(privkeys(e)) \subseteq F_e^i(privkeys(e))$, and consequently, $FIX(F_e) = F_e^{n_1}(privkeys(e)) = \bigcap_i F_e^i(privkeys(e))$. □

Suppose that $e \in Pat$ and σ is a key bijection function (obviously the key function has the property that if $\sigma(k) = k_1$, then $\sigma(k^{-1}) = k_1^{-1}$). By $e\sigma$ we denote the pattern obtained from e by replacing each key k in e by $\sigma(k)$. The adversarial knowledge set can be defined with respect to an *inductive approach* (e.g. [3]) or a *co-inductive approach* (e.g. [27]), which corresponds to the least fixed point and the greatest fixed point of the key-recovery function, respectively. That is, the set of private keys that the formal adversary can obtain from e is defined to be $fix(F_e)$ in the inductive setting, and $FIX(F_e)$) in the co-inductive setting. We can now define the *pattern of expressions* in both the inductive and co-inductive settings. For $e \in Pat$, $pat_I(e) = p(e, fix(\mathcal{F}_e))$ and $pat_C(e) = p(e, FIX(\mathcal{F}_e))$. We are now ready to define the notion of symbolic equivalence of expressions: we say two patterns e_1 and e_2 are co-inductively equivalent, written as $e_1 \cong_c e_2$, if there exists a key bijection function σ under which $pat_C(e_1) = pat_C(e_2)\sigma$. Similarly, we say that e_1 and e_2 are inductively equivalent, written as $e_1 \cong_I e_2$, if there exists a key bijection function σ under which $pat_I(e_1) = pat_I(e_2)\sigma$. Throughout the paper, we take the co-inductive definition of equivalence, and we will write it as $\cong$.

We can generalize the pattern definition when the adversary has *a priori* information about T. For this purpose, denote by eT the pattern obtained from e by pairing it with all keys in T. So if $T = \{K_1, \ldots, K_l\}$, then $eT = (e, K_1, \ldots, K_l)$. We now define $pat_C(e, T) = p(e, FIX(\mathcal{F}_{eT}))$ (for the case of induction, we define $pat_I(e, T) = p(e, fix(\mathcal{F}_{eT}))$). Note that we clearly have $pat_C(e) = pat_C(e, \emptyset)$. The

following fact can be easily verified:

$pat_C(e_1, T) = pat_C(e_2, T) \Leftrightarrow pat_C(e_1 T) = pat_C(e_2 T)$

Note that the above fact does not hold if we replace $=$ with $\cong$. For example, let $m = (\{0\}_{k_1}, \{1\}_{k_2})$ and $n = (\{0\}_{k_2}, \{1\}_{k_1})$, and $T = \{k_1^{-1}, k_2^{-1}\}$. We have $pat(m, T) \cong pat(n, T)$, $pat(mT) = (\{0\}_{k_1}, \{1\}_{k_2}, k_1^{-1}, k_2^{-1})$, and $pat(nT) = (\{0\}_{k_2}, \{1\}_{k_1}, k_1^{-1}, k_2^{-1})$. Now it is obvious that $pat(mT) \ncong pat(nT)$ (they are not equivalent even upto key-renaming).
Proofs of the following properties can be easily obtained:

Proposition 1. *1.* $privkeys(p(e, T)) = \mathcal{F}_e(T)$

2. $p(p(e, T_1), T_2) = p(e, T_1 \cap T_2)$

3. $privkeys(p(e, T)) \subseteq privkeys(e)$

4. $FIX(F_e) = FIX(F_{e_1})$, *where* $e_1 = p(e, privkeys(e))$

Let $e \in Pat$. We say that public key k_1 encrypts private key k^{-1} in e, if there exists a pattern e_1 such that $k^{-1} \in privkeys(e_1)$ and $\{e_1\}_{k_1} \sqsubseteq e$. To every pattern e, we can associate an underlying key graph $G_e = (V_e, E_e)$ as follows:

$$V_e = \{(k, k^{-1}) \mid k \in pubkeys(e) \text{ or } k^{-1} \in privkeys(e))\}$$
$$E_e = \{((k, k^{-1}), (k_1, k_1^{-1}) \mid k \text{ encrypts } k_1^{-1} \text{ in } e\}.$$

A pattern e is *encryption cyclic* (*encryption acyclic*) if its underlying key graph is cyclic (acyclic). In the following lemma, we establish the relation between $FIX(F_e)$ and $fix(F_e)$. In particular, we show that as long as e is acyclic, we have $fix(F_e) = FIX(F_e)$. Note that the converse of this fact is not valid in general. For example if $e = (\{k^{-1}\}_k, k^{-1})$, then e is cyclic but $fix(F_e) = FIX(F_e)$. We remark that a symmetric version of this fact was proved in [27]. However, we can give an easier proof for the asymmetric case using some basic facts from graph theory.

Lemma 2. *Suppose e is an acyclic pattern. We have:* $fix(\mathcal{F}_e) = FIX(\mathcal{F}_e)$.

Proof. Assuming that $fix(F_e) \neq FIX(F_e)$, we prove that the graph G_e is cyclic. Let $T_1 = fix(F_e)$, $T_2 = FIX(F_e)$, and $T = T_2 - T_1$. Note that for every $k^{-1} \in T$, there exists another $k_1^{-1} \in T$ such that k_1 encrypts k^{-1}, because otherwise k^{-1} were in T_1 as well. Therefore, if we let G_1 be the induced subgraph of G_e on the vertices

corresponding to T, we can see that every vertex in G_e has an in-degree of at least one. It is a well-known fact in graph theory that if every vertex of a directed graph G has an in-degree of at least one, then G is cyclic. Therefore G_e is cyclic and the proof is complete.

□

Given a pattern e and a set of private keys T, we define the set of derivable sub-terms of e which are undecryptable with respect to T. We need this definition in the proof of our soundness results.

Definition 1. *Let* $T \subseteq K_{priv}$, *and* $e \in Pat \cup \{struct(e) | e \in Exp\}$. *We define* $undec_e(T)$, *as follow:*

- *if* $e = (e_1, e_2)$, $undec_e(T) = undec_{e_1}(T) \cup undec_{e_2}(T)$,
- *if* $e = \{e_3\}_K$ *and* $K^{-1} \in T$, $undec_e(T) = undec_{e_3}(T)$,
- *if* $e = \{e_3\}_K$ *and* $K^{-1} \notin T$, $undec_e(T) = \{\ \{e_3\}_K\ \}$,
- *if* $e \neq (e_1, e_2)$ *and* $e \neq \{e_3\}_K$ *then* $undec_e(T) = \emptyset$

Chapter 3

Computational Encryption

3.0.7 Standard Definitions of Computational Security

Computational treatment of encryption takes a less abstract view of cryptography than the formal treatment. In this model, messages are no longer syntactic objects, rather they are finite bit-strings chosen from some distribution. Encryption and other cryptographic primitives are formalized as *probabilistic polynomial-time (PPT)* algorithms. The computational adversary is a probabilistic polynomial-time Turing machine which is able to perform any polynomial time computation during its execution, as opposed to formal adversaries which are confined to fixed certain rules for computation. We start this chapter by introducing the syntax of asymmetric encryption schemes:

A *public key encryption scheme* is a tuple $\Pi = (Gen, Enc, Dec)$ of probabilistic polynomial time algorithms, all of which take as input a string η in unary called the *security parameter*. Technically, the security parameter is present to measure the amount of security that the system provides. The other components are:

- The *key generation algorithm Gen* takes as input the *security parameter* η and outputs a pair of public/private keys (pk, sk), written as $(pk, sk) \leftarrow Gen(1^\eta)$. As the function might be probabilistic (which is usually the case), we use $\leftarrow$ to denote the randomness involved. We denote by $Gen_i(.)$ the ith component of $Gen(.)$ function, so that $Gen(1^\eta) = (Gen_1(1^\eta), Gen_2(1^\eta))$

- The encryption algorithm Enc takes as input a public key pk and a plaintext $m \in \{0,1\}^*$, and outputs a ciphertext $c \leftarrow Enc(pk, m)$. We may sometimes adopt a more convenient notation and write $Enc(pk, .)$ as $Enc_{pk}(.)$

- The decryption algorithm Dec takes as input a private key sk and a ciphertext c, and outputs the decryption of the ciphertext c under key sk. If (pk, sk) is a pair of keys output by Gen, we require that $Dec_{sk}(Enc_{pk}(m)) = m$. We assume, without loss of generality, that the decryption algorithm is deterministic.

Suppose $\vec{M} = (m_1, \ldots, m_n)$ is a vector of plaintexts and k is a key. Then $Enc(\vec{M}, k)$ denotes $(Enc(m_1, k), \ldots, Enc(m_n, k))$. We write $x \leftarrow^R S$ to denote choosing an element x uniformly at random from set S. If $\mathcal{D}$ is a distribution, we let $supp(\mathcal{D})$ denote the *support* of $\mathcal{D}$.
Security assertions about encryption schemes are typically formalized as asymptotic statements: an encryption scheme is secure if the probability that an adversary can do something "unfavorable" (to be formalized later) is negligible with respect to the security parameter:

Definition 2. *(Negligible function) A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is said to be* negligible *if for any $c > 0$, there exists n_0 such that for all $n > n_o$ we have*

$$\epsilon(n) < \frac{1}{n^c}$$

We may sometimes write $negl(.)$ to denote a function which is negligible. We present three well-known notions of computational security, called *IND-CPA*, *IND-CCA1*, *IND-CCA2*, in order of increasing strength. Here IND represents the security goal, *indistinguishability of encryptions* due to Goldwasser and Micali [19], which formalizes the adversary's inability to tell the difference between encryptions of (its own chosen) plaintexts. All these notion are formalized based on an *indistinguishability experiment* expressed as a game in which the adversary is involved and tries to win. The idea is as follow: first a random public key pk is chosen whose private key is kept as secret. Then the adversary is encouraged to submit two candidates of plaintexts (of equal length), which the adversary thinks it can distinguish between their encryptions. Then one of these two messages is chosen at random, and its encryption under pk, called *challenge ciphertext*, is given to the adversary. Now the adversary is challenged to determine which one was encrypted. Under CPA attack model, i.e. *chosen plaintext-attack*, the adversary is just given the public key pk, which enables the adversary to obtain ciphertexts of plaintexts of its own choice. Under CCA1 attack model, *non-adaptive chosen ciphertext attack*, besides pk, the adversary is provided with a decryption oracle which decrypts ciphertexts with respect to pk. However the

adversary is allowed to use this oracle as long as it has not submitted its plaintext candidates. Finally, under CCA2 attack model, *adaptive chosen-ciphertext attack*, the adversary continues to have access to the decryption oracle after receiving the challenge ciphertext, with the only restriction that it may not ask for decryption of the challenge ciphertext. For simplicity and conciseness, we define the experiment for all three notions together: (see e.g. [18] for more discussion on these notions of security)

Single message indistinguishability experiment for IND-atk

Given a public key encryption scheme $\Pi = (Gen, Enc, Dec)$, security parameter η, and atk $\in \{cpa,\ cca1,\ cca2\}$, we define the "single message indistinguishability experiment for IND-atk" as follow:

1. $(pk, sk) \leftarrow G(1^\eta)$
2. The adversary $\mathcal{A}$ is given pk and access to the oracle $D_1^{atk}(.)$, where $D_1^{atk}(x) = Dec(x, sk)$ if atk $\in \{cca1,\ cc2\}$, and $D_1^{atk}(x) = \epsilon$ (i.e. empty oracle) if $atk = cpa$. It finally outputs two messages m_0 , m_1 of the same length.
3. A bit b is chosen at random from $\{0, 1\}$ and the ciphertext $c \leftarrow Enc(m_b, pk)$ is computed and given to $\mathcal{A}$.
4. If $atk = cca2$, $\mathcal{A}$ is given access to the oracle $D_2(.)$, where $D_2(x) = Dec(x, sk)$ if $x \neq c$ and $D_2(x) = \perp$ if $x = c$. It finally outputs some bit g.
5. the probability of success is defined with respect to the probability that $b = g$.

Definition 3. *For atk* $\in \{cpa,\ cca1,\ cca2\}$*, an encryption scheme* $\Pi = (Gen, Enc, Dec)$ *is said to provide indistinguishability of single encryption under IND-atk attack (or IND-atk secure), if for all PPT adversaries* $\mathcal{A}$ *the probability of success in the above experiment is* $\frac{1}{2} + negl(\eta)$.

Chosen-plaintext security ensures that the adversary is not able to distinguish between the encryptions of two plaintexts of its own choice. In the next chapter, we show that this notion actually gives rise to a stronger form of indistinguishability, called *indistinguishability of expressions*. That is, we show if two symbolically equivalent expressions are evaluated under a CPA-secure encryption system (to be defined later), the resulting computational messages will be indistinguishable to any computational adversary. For our results, however, we need to develop a generalized

version of indistinguishability experiment where the adversary will have the possibility to submit multiple messages. In particular, our proposed experiment is run under a number of keys rather than just a single key, and it lets the adversary submit multiple messages for encryption. We call this new experiment *multiple message indistinguishability experiment*. It is not hard to see, using a standard hybrid argument, that, for all attack models we mentioned earlier, the multiple-message-based definition of security is equivalent to the standard one (i.e. single-message-based). We remark that a somewhat similar generalization has also been considered in [8]:

Multiple message indistinguishability experiment

Given a public key encryption scheme $\Pi = (Gen, Enc, Dec)$, security parameter η, and atk $\in \{cpa,\ cca1,\ cca2\}$, we define the *multiple message indistinguishability experiment* for IND-atk as follow:

1. $G(1^\eta)$ is run r times to produce r pairs of public/private keys: $(pk_1, sk_1) \leftarrow G(1^\eta)\ ;\ \ldots\ ;\ (pk_r, sk_r) \leftarrow G(1^\eta)$

2. The adversary $\mathcal{A}$ is given $(pk_1, \ldots, pk_r)$ and access to the set of oracles $\{O_i^1(.)\}_{1\leq i\leq r}$, where $O_i^1(x) = Dec(x, sk_i)$ if atk $\in \{cca1,\ cca2\}$, and $O_i^1(x) = \epsilon$ if atk = cpa. For each key pk_i, $\mathcal{A}$ selects two message vectors $\vec{M}_i = (m_i^1, \ldots, m_i^{p_i})$ and $\vec{N}_i = (n_i^1, \ldots, n_i^{p_i})$, with the restriction that $|m_i^j| = |n_i^j|$ for all $1 \leq j \leq p_i$. Finally $\mathcal{A}$ outputs two vectors of vectors: $\vec{M} = (\vec{M}_1, \ldots, \vec{M}_r)$ and $\vec{N} = (\vec{N}_1, \ldots, \vec{N}_r)$.

3. A random bit $b \in \{0, 1\}$ is chosen, and the challenge ciphertext $\vec{C} = \vec{B}_b$ is given to $\mathcal{A}$, where: $\vec{B}_0 = (enc(\vec{M}_1, pk_1), \ldots, enc(\vec{M}_r, pk_r))$ and $\vec{B}_1 = (enc(\vec{N}_1, pk_1), \ldots, enc(\vec{N}_r, pk_r))$.

4. Let $\vec{C} = (\vec{C}_1, \ldots, \vec{C}_r)$ be the challenge ciphertext. If atk = cca2, $\mathcal{A}$ is given access to the set of oracles $\{O_i^2(.)\}_{1\leq i\leq r}$, where $O_i^2(x) = Dec(x, sk_i)$ if $x \notin \vec{C}_i$, and $O_i^2(x) = \perp$ if $x \in \vec{C}_i$. It finally outputs some bit g.

5. the probability of success is defined with respect to the probability that $b = g$.

Definition 4. *For atk $\in \{cpa,\ cca1,\ cca2\}$, a public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ provides* indistinguishability of multiple encryptions under IND-atk attack *(or* IND-atk secure *) if for all PPT adversaries $\mathcal{A}$, the probability of success in the above experiment is negligible.*

As mentioned before, one can prove that under any attack-model presented, an encryption scheme is secure with respect to single-message-based definition if and only if it is secure with respect the multiple-message-based one.

3.0.8 Interpreting Formal Expressions in the Computational World

In chapter 2, we gave the formal semantics of expressions in our language, and we showed how this semantics leads to a notion of equivalence in the formal setting. In this chapter, we give a second semantics for our language which treats expressions in a computational sense, providing a more concrete interpretation of them. Our semantics is based on the semantics introduced in [21] for expressions with asymmetric encryption. As the first step for defining the *computational semantics*, we need to fix a *computational pairing function* $\langle ., . \rangle$, which serves as the computational counterpart of the paring operator by concatenating two bit-strings:

$$\langle ., . \rangle : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

Obviously, the pairing function must be one-to-one. As a notational convention, we write $\langle x_1,\ x_2, \ldots x_n \rangle$ to mean $\langle \ldots \langle \langle x_1,\ x_2 \rangle,\ x_3 \rangle \ldots,\ x_n \rangle$. Now given a public-key encryption scheme Π, we can present a *computational encoding* which maps a given pattern into an *ensemble*, that is a family of computational distributions indexed by the security parameter. More precisely, for each choice of a security parameter, each pattern e in the formal setting induces a computational distribution. The way that the mapping operates is very natural and is presented below. We first show how we can map formal expressions (i.e. member of Exp) into probability distributions, and then we extend the encoding function to include the computational interpretations of patterns.

Definition 5. *Let $\Pi = (Gen, Enc, Dec)$ be a public-key encryption scheme and η be a security parameter. Let τ be a* key assignment function *which assigns a random key value to each key symbol. We begin by setting $(\tau(K), \tau(K^{-1})) \leftarrow Gen(\eta)$ for each key K. We define $[e]_{\Pi}^{\eta,\tau}$,* the computational encoding *of the expression e with respect to Π, η, and τ, as follow:*

- $e \in \mathcal{K}_{priv}$*:* $[e]_{\Pi}^{\eta,\tau} = \langle \tau(e),$ *"privkey"*$\rangle$

- $e \in \mathcal{K}_{pub}$: $[e]_{\Pi}^{\eta,\tau} = \langle \tau(e), \text{“pubkey”} \rangle$
- $e \in Block$: $[e]_{\Pi}^{\eta,\tau} = \langle e', \text{“block”} \rangle$, *where* $|e'| = r$ *for some fixed* r
- $e = (e_1, e_2)$: $[(e_1, e_2)]_{\Pi}^{\eta,\tau} = \langle [e_1]_{\Pi}^{\eta,\tau}, [e_2]_{\Pi}^{\eta,\tau}, \text{“pair”} \rangle$
- $e = \{e_1\}_K$: $[e]_{\Pi}^{\eta,\tau} = \langle Enc([e_1]_{\Pi}^{\eta,\tau}, \tau(K)), \tau(K), \text{“ciphertext”} \rangle$

As it can be seen, every formal expression is mapped into a family of computational distributions indexed by the security parameter. For a fixed security parameter η, the computational distribution of an expression e depends on the randomness involved in both the key-generation algorithm and the encryption algorithm. The inclusion of the underlying key in the computational image of encryptions is due to the fact that in public-key cryptography, an encryption does not hide its public key ([21]). In order to define the computational interpretation of patterns, we need to give a computational meaning to $struct(e)$, where $e \in Exp$. We do so by assigning $[struct(e)]_{\Pi}^{\eta,\tau} = 0^{|E|}$ where $E \in supp[e]_{\Pi}^{\eta,\tau}$. In order for this mapping to be well-defined, we will require that for all formal terms e_1 and e_2 , if $struct(e_1) = struct(e_2)$ then $|[e_1]| = |[e_2]|$. This condition can be guaranteed by requiring that the key-generation function, encryption function, and the pairing function $\langle\rangle$ used in the above definition be length-regular (a function f is *length-regular*, if for every $x, y \in \{0,1\}^*$ we have: $|x| = |y| \Rightarrow |f(x)| = |f(y)|$). Also, we assume that all elements of $Block$ are mapped to bit-strings of the same length, and none of them is mapped to the string of all zeros.

As one more assumption, for all pk's output by the key generation algorithm, if $x \in \{0,1\}^*$ is in the plaintext space of Enc_{pk}, we assume that all strings in $\{0,1\}^{|x|}$ are also in the plaintext space of Enc_{pk}. This automatically implies that for all x in the plaintext space of Enc_{pk}, we have $|Enc_{pk}(x)| \geq |x|$. If $E = \{e_1, e_2, ..., e_k\}$ is a set of patterns, we define $[E]_{\Pi}^{\eta,\tau} = \langle [e_1]_{\Pi}^{\eta,\tau}, [e_2]_{\Pi}^{\eta,\tau} \ldots [e_k]_{\Pi}^{\eta,\tau} \rangle$. We also introduce the following notation: suppose $E \in supp[e]_{\Pi}^{\eta,\tau}$, and $e_1 \sqsubseteq e$. We denote by $E_{[e_1]_{\Pi}^{\eta,\tau}}$ the corresponding computational image of e_1 in E (using Π and τ, it is defined in a straightforward way). In cases where e_1 occurs more than once as a subexpression in e (for example, $e = (e_1, e_1)$), it will be clear from the context to which occurrence $E_{[e_1]_{\Pi}^{\eta,\tau}}$ is referring to. We may generalize this notation as follow: suppose Q is a set of patterns such that each of them is a subexpression of e, we define $E_{[Q]_{\Pi}^{\eta,\tau}} = \{E_{[e]_{\Pi}^{\eta,\tau}} \mid e \in Q\}$. When η, τ, and Π are clear from the context, we may write $E_{[e_1]}$ for $E_{[e_1]_{\Pi}^{\eta,\tau}}$. Finally, if pk is a public key value, we denote any corresponding private key value by pk^{-1}, meaning that $(pk, pk^{-1}) \in Gen(1^{\eta})$.

3.0.9 Computational Indistinguishability of Expressions

Let T be a set of public-key values, and s be the computational image of some expression. We define the *computational derivability relation* $\vdash^T$ as the least binary relation with the following properties:

1. $s \vdash^T s$
2. if $s \vdash^T \langle s_1, s_2, \text{“pair”} \rangle$ then $s \vdash^T s_1$ and $s \vdash^T s_2$.
3. if $s \vdash^T \langle c, pk, \text{“ciphertext”} \rangle$ and $pk \in T$, then $s \vdash^T Dec_{pk^{-1}}(s)$.

Having defined the derivability relation, we form the set of all ciphertexts *visible* in s relative to T, denoted by $Vis_s(T)$, as follow: if $s \vdash^T \langle c, pk, \text{“ciphertext”} \rangle$, add $\langle c, pk, \text{“ciphertext”} \rangle$ to $Vis_s(T)$.

In the following, we present our notion of computational equivalence, which is a stronger form of the *standard notion of computational indistinguishability* [17], which is the typical definition of "similarity" in computational cryptography. In principle, the standard notion of computational indistinguishability relates two families of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ if they look the same to every efficient adversary $\mathcal{A}$, formulated as:

$$\Pr[\mathcal{A}(1^n, x \leftarrow X_n) = 1] - \Pr[\mathcal{A}(1^n, y \leftarrow Y_n) = 1] = negl(n)$$

in the vocabulary of complexity theory. Our notion of computational equivalence for expressions is a stronger version of computational indistinguishability, where the adversary is provided with a decryption oracle. The original characterization of this notion was formulated in the inductive setting [21]. We give here both an inductive and co-inductive characterization of this notion, and discuss their differences:

Definition 6. *(Coinductive strong computational indistinguishability) Let T and T_1 be two sets of public key symbols. Given a public key encryption scheme Π, we say that $[e_1]_\Pi^{\eta,\tau} \approx_C^{O_x^{T_1,T}} [e_2]_\Pi^{\eta,\tau}$, if for all PPT adversaries $\mathcal{A}$ and random key assignment function τ:*

$$\Pr[d_1 \leftarrow [e_1]_\Pi^{\eta,\tau} : \mathcal{A}^{O_{d_1}^{T_1,T}}(1^\eta, d_1) = 1] - \Pr[d_2 \leftarrow [e_2]_\Pi^{\eta,\tau} : \mathcal{A}^{O_{d_2}^{T_1,T}}(1^\eta, d_2) = 1]$$

is negligible, where $O_{d_i}^{T_1,T}(\sigma, pk)$ returns $Dec_{pk^{-1}}(\sigma)$ if

1. *either* $pk = \tau(k)$ *for some* $k \in T$, *or*

2. $pk = \tau(k)$ *for some* $k \in T_1$ *and* $\langle \sigma, pk, \text{"ciphertext"} \rangle \notin Vis_{d_i}(\tau(FIX^{-1}(\mathcal{F}_{e_i T^{-1}})))$,

and returns $\perp$ *otherwise.*

Henceforth, for convenience, we write $\approx_C^{T_1,T}$ for $\approx_C^{O_x^{T_1,T}}$. We typically take $T_1 = pubkeys(e_1) \cup pubkeys(e_2)$. However, sometimes we may have to consider the general case where $T_1 \neq pubkeys(e_1) \cup pubkeys(e_2)$. The above indistinguishability notion is defined based on a family of oracles $O_x^{T_1,T}$, where on input d, the adversary's granted oracle is $O_d^{T_1,T}$. We now give the intuition behind the definition. Our intuition is that if Π satisfies sufficiently strong security conditions, then we may have $[e]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),T} [pat_C(e,T)]_\Pi^{\eta,\tau}$. Let's see what the decryption oracle does here: first of all, since the formal adversary, given set T^{-1} for decryption, is not able to distinguish between e and $pat_C(e,T)$, we expect that if we allow the computational adversary's oracle to decrypt with respect to keys in $\tau(T)$, the computational adversary will not be able to tell whether its input E was drawn from $[e]_\Pi^{\eta,\tau}$ or $[pat_C(e,T)]_\Pi^{\eta,\tau}$ (condition 1 of the above definition). Moreover, the set of ciphertexts which differ between e and $pat_C(e,T)$ is $undec_e(FIX^{-1}(\mathcal{F}_{eT^{-1}}))$, whose elements are transformed into blobs in $pat_C(e,T)$. In the computational setting, we can think of the corresponding computational values of elements of $undec_e(FIX^{-1}(\mathcal{F}_{eT^{-1}}))$ in E as *challenge ciphertexts*, where the decryption of any of those will indicate whether $E \in supp[e]_\Pi^{\eta,\tau}$, or $E \in supp[pat_C(e,T)]_\Pi^{\eta,\tau}$. On the other hand, the computational adversary, being able to decrypt with respect to keys in $\tau(T)$, may be able to obtain challenge ciphertexts accordingly. Thus we expect as long as the oracle does not decrypt challenge ciphertexts, the computational adversary should not be able to tell whether E was drawn from $[e]_\Pi^{\eta,\tau}$ or $[pat_C(e,T)]_\Pi^{\eta,\tau}$ (condition 2 of the above definition). However, for convenience, as in [21], we define a larger set of challenge ciphertexts which includes all ciphertexts co-inductively derivable from E using set $\tau(T)$ for decryption. Note that this new set also contains those ciphertexts which correspond to "non-blobs" in $pat_C(e,T)$; the decryption oracle still does not decrypt those ciphertexts, but this is not a big restriction as most of them can already be decrypted by the adversary itself.

Two remarks about definition 6 are in order:

Remark 1. *In general, if* $T_1 \subset T_2$, $[e_1]_\Pi^{\eta,\tau} \approx_C^{T_1,T} [e_2]_\Pi^{\eta,\tau}$ *does not imply* $[e_1]_\Pi^{\eta,\tau} \approx_C^{T_2,T} [e_2]_\Pi^{\eta,\tau}$, *or vice versa. For this to hold, one has to prove that if* d *is a sample from* $[e_1]_\Pi^{\eta,\tau}$ *or* $[e_2]_\Pi^{\eta,\tau}$, *then* (σ, pk) *is a valid query to* $O_d^{T_1,T}$ *iff* (σ, pk) *is valid to* $O_d^{T_2,T}$,

except with negligible probability. As a special case, it can, however, be seen that if $T_1 = pubkeys(e_1) \cup pubkeys(e_2) \subset T_2$, *and* $privkeys^{-1}(e_1) \cup privkeys^{-1}(e_2) \subseteq pubkeys(e_1) \cup pubkeys(e_2)$ *then:*

$$[e_1]_\Pi^{\eta,\tau} \approx_C^{T_1,T} [e_2]_\Pi^{\eta,\tau} \Leftrightarrow [e_1]_\Pi^{\eta,\tau} \approx_C^{T_2,T} [e_2]_\Pi^{\eta,\tau}$$

In order to see this, note that the only type of queries that may be answered by $O_d^{T_2,T}$, *but not by* $O_d^{T_1,T}$ *is* (σ, pk) *for* $pk = \tau(k)$ *for* $k \in T_2 - T_1$. *But then the value of* pk *would be completely independent of the sample* d *given to the adversary, because neither of the key symbols* k *and* k^{-1} *appear in* e_1 *or in* e_2, *so the probability that an adversary computes* $\tau(k)$ *is negligible (of course, this relies on the assumption that for* $pk \in \{0,1\}^*$: $pr[Gen_1(1^\eta) = pk] = negl(\eta)$, *which is the case for all encryption schemes we consider in this paper).*

Remark 2. *(Inductive version of strong computational indistinguishability) In definition 6, if we replace* $\sigma \notin Vis_{d_i}(\tau(FIX^{-1}(\mathcal{F}_{e_i T^{-1}})))$ *with* $\sigma \notin Vis_{d_i}(\tau(fix^{-1}(\mathcal{F}_{e_i T^{-1}})))$, *we obtain an inductive version of strong indistinguishability, notation* $\approx_I^{T_1,T}$, *which is equivalent to the characterization of [21]. The result of [21] indicates that in the absence of key-cycles, if* $pubkeys(e_1) = pubkeys(e_2)$ *and* Π *is CCA-2 secure, then* $pat_I(e_1, T) = pat_I(e_2, T)$ *implies* $[e_1]_\Pi \approx_I^{pubkeys(e_1),T} [e_2]_\Pi$.

Motivated by the above discussion, we derive a strong notion of computational security which captures the assumptions made about the strength of formal adversaries in distinguishing between expressions:

Definition 7. *We say that a public key encryption scheme* Π *provides strong co-inductive (resp. inductive) public-key indistinguishability if for all* $m \in Pat$, *and finite* $T \subseteq K_{pub}$*:*

$$[m]_\Pi^{\eta,\tau} \approx_{\mathcal{X}}^{pubkeys(m),T} [pat_X(m, T^{-1})]_\Pi^{\eta,\tau}$$

where $\mathcal{X} = C$ *(resp.* $\mathcal{X} = I$*).*

In the above definition, if we drop the decryption oracle access, we will obtain a weaker form of indistinguishability-based notion of security as follow:

Definition 8. *We say that a public key encryption scheme* Π *provides weak co-inductive (resp. inductive) public-key indistinguishability if for all* $m \in Pat$, *and finite* $T \subseteq K_{pub}$*:*

$$[m]_\Pi^{\eta,\tau} \approx [pat_X(m)]_\Pi^{\eta,\tau}$$

where $\mathcal{X} = C$ *(resp.* $\mathcal{X} = I$*).*

3.0.10 Non-Malleability of Expressions

The notion of strong public-key indistinguishability characterizes the computational difficulty of distinguishing between computational expressions. Another approach to comparing formal and computational adversaries was formulated in [21], where the goal of the adversary is not to distinguish between two expressions, but to transform one of them to another. More specifically, in a Dolev-Yao-style model, it is assumed that at any time the formal adversary is able to perform certain operations, namely decryption with respect to adversary's keys or the keys that have been already revealed as part of the protocol, encryption with public keys, pairing two elements, and separation of a pair into two elements. Based on this assumption, we can define the notion of the *closure* of an expression, where $Closure(e)$ represents the set of expressions that the formal adversary can produce from e. Computational soundness in this setting means that the computational adversary, when given the computational interpretation of e, has a negligible chance of producing the computational interpretation of any expression outside $Closure(e)$. The original characterization of [21] for non-malleability was based on an inductive definition. However, adapted to our framework, we give a co-inductive definition of closure as follow:

Definition 9. *(co-inductive closure) Let* S *be a set of formal expressions. The co-inductive closure of* S*, written* $Closure_C(S)$*, is the smallest set satisfying the following properties:*

- $S \cup K_{pub} \cup Block \subseteq Closure_C(S)$,
- $FIX(\mathcal{F}_S) \subseteq Closure_C(S)$,
- *if* $\{e\}_k \in Closure_C(S)$ *and* $k^{-1} \in Closure_C(S)$*, then* $e \in Closure_C(S)$,
- *if* $e \in Closure_C(S)$ *and* $k \in Closure_C(S)$*, then* $\{e\}_k \in Closure_C(S)$,
- *if* $e_1 \in Closure_C(S)$ *and* $e_2 \in Closure_C(S)$*, then* $(e_1, e_2) \in Closure_C(S)$*, and*
- *if* $(e_1, e_2) \in Closure_C(S)$*, then* $e_1 \in Closure_C(S)$ *and* $e_2 \in Closure_C(S)$.

In line 2 of the above definition, if we replace $FIX(\mathcal{F}_S) \subseteq Closure(S)$ with $fix(\mathcal{F}_S) \subseteq Closure(S)$, we obtain an inductive version of closure, which we denote

by $Closure_I(S)$. In [21], a computational version of inductive closure is given, called *Dolev-Yao public-key non-malleability* , and it is proved in the absence of key-cycles, if an encryption scheme provides inductive public-key indistinguishability(definition 7), it also provides inductive Dolev-Yao public-key non-malleability. Adapted to our framework, with a non-malleability characterization based on co-inductive closure, we can extend the non-malleability result of [21] in the presence of key-cycles. Let's first walk through the idea of Dolev-Yao public-key non-malleability, which is a computational version of definition 9. We refer the reader to the original paper [21] for motivation and intuition behind the definition. Suppose S is a set of formal expressions and $e \notin Closure(S)$. Very informally, we say that an encryption scheme Π provides Dolev-Yao public-key non-malleability if the probability that any adversary $\mathcal{A}$ when given $E_1 \leftarrow [S]_{\Pi}^{\eta,\tau}$ outputs E_2 where E_2 is a possible computational encoding of e (i.e. $E_2 \in supp[e]_{\Pi}^{\eta,\tau}$) is negligible. Recall that, in definition 9, we assumed that the formal adversary is in possession of all formal public key symbols. In order to translate this in the computational world, since the length of the set K_{pub} may be infinite, we cannot feed the computational values of all public key symbols as an input to the adversary (because it will receive an input of infinite length in case that $\mathcal{K}_{pub}$ is infinite!). Instead, we give the adversary an access to oracle $pbK_{\eta}^{\tau}(.)$ where $pbK_{\eta}^{\tau}(k)$ returns $\tau(k)$. We are now ready to see the formal definition:

Definition 10. *(Coinductive Dolev-Yao public-key non-malleability) The encryption scheme* $\Pi = (Gen, Enc, Dec)$ *provides* co-inductive (resp. inductive) Dolev-Yao public-key non-malleability *if for all* $S \subseteq Exp$, $e \notin Closure_C(S)$ *(resp.* $e \notin Closure_I(S)$ *), and PPT adversaries* $\mathcal{A}$*, the following function:*

$$\epsilon(\eta) = \Pr[\ E_1 \leftarrow [S]_{\Pi}^{\eta,\tau};\ E_2 \leftarrow \mathcal{A}^{pbK_{\eta}^{\tau}(.)}(1^{\eta}, E_1) :\ E_2 \in supp[e]_{\Pi}^{\eta,\tau}\ ]$$

is negligible; where $pbK_{\eta}^{\tau}(k)$ *returns* $\tau(k)$.

Chapter 4

Relating the Two Views

4.1 Indistinguishability-Computational Soundness

In this chapter, we explore the computational soundness of the co-inductive definition of formal equivalence in the presence of key-cycles. Henceforth, as a shorthand, we will denote the IND-CCA2 security notion as CCA-2 security. [1] The main result of this chapter (corollary 1) states that if Π provides CCA-2 security, it also provides co-inductive strong indistinguishability in the presence of key-cycles, extending the result of [21]. That is, we show that if two expressions are co-inductively equivalent, then their computational evaluations under CCA-2 secure encryption schemes result in strongly indistinguishable ensembles.

We first give some remark about definition 6: it can be easily seen from this definition that if e_1 and e_2 are two patterns and $T_1 = pubkeys(e_1) \cup pubkeys(e_2)$ then:

$$[e_1 T T^{-1}]_{\Pi}^{\eta,\tau} \approx_C^{T_1,\emptyset} [e_2 T T^{-1}]_{\Pi}^{\eta,\tau} \Rightarrow [e_1]_{\Pi}^{\eta,\tau} \approx_C^{T_1,T} [e_2]_{\Pi}^{\eta,\tau}$$

In the rest of this chapter, when we want to prove $[e_1]_{\Pi}^{\eta,\tau} \approx_C^{T_1,T} [e_2]_{\Pi}^{\eta,\tau}$, we usually prove the stronger result $[e_1 T T^{-1}]_{\Pi}^{\eta,\tau} \approx_C^{T_1,\emptyset} [e_2 T T^{-1}]_{\Pi}^{\eta,\tau}$.

Theorem 1. *Suppose $\Pi = (Gen, Enc, Dec)$ is CCA-2 secure, and $e \in Pat$. Letting $T = privkeys(e)$, we have*

$$[e T T^{-1}]_{\Pi}^{\eta,\tau} \approx_C^{pubkeys(e),\emptyset} [p(e,T) T T^{-1}]_{\Pi}^{\eta,\tau}.$$

[1] One reason for this is that it has been proved that under the two well-known adversarial goals, i.e. indistinguishability of encryptions and *non-malleability* of [15], the CCA2 attack model gives rise to equivalent security notions, i.e. NM-CCA2 $\Leftrightarrow$ IND-CCA2

Proof. Letting $e_1 = eTT^{-1}$ and $e_2 = p(e,T)TT^{-1} = p(e_1, privkeys(e_1))$, suppose, for the sake of contradiction, that there exists an adversary $\mathcal{A}^{pubkeys(e),\emptyset}$ which can distinguish between $[e_1]^{\eta,\tau}_{\Pi}$ and $[e_2]^{\eta,\tau}_{\Pi}$. We use $\mathcal{A}^{pubkeys(e),\emptyset}$ to construct an adversary $\mathcal{B}$ to break the CCA-2 security of Π (definition 4). The adversary $\mathcal{B}$ works as follow: for every private key $k^{-1} \in T$ (remember that $T = privkeys(e_1)$), it runs the key generation algorithm to obtain a pair of key values $(\tau(k), \tau(k^{-1}))$ for (k, k^{-1}), and adds $\tau(k)$ to the set $priv\_known$. Letting $R = pubkeys(e) - T^{-1} = \{k_1, \ldots, k_{|R|}\}$ (note that R is the set of public key symbols whose private keys do not occur in e), we assume that the multiple message indistinguishability experiment that $\mathcal{B}$ is involved in is run under $|R|$ randomly chosen public keys $pk_1, \ldots, pk_{|R|}$, and thus we can think of pk_i as a public key value for k_i (i.e. we can take $\tau(k_i) = pk_i$). Letting $priv\_unknown = \{pk_1, ..., pk_{|R|}\}$, now for each $pk_i \in priv\_unknown$, $\mathcal{B}$ creates two vectors of messages $\vec{M}_i$ and $\vec{N}_i$, which represent the set of messages that $\mathcal{B}$ submits to be encrypted under key pk_i. Let $undec_e(privkeys(e))$ be the set of undecryptable messages of e with respect to $privkeys(e)$ (see definition 1). For all $m \in undec_e(privkeys(e))$, if $m = \{r\}_{k_i}$, $\mathcal{B}$ generates two bit-strings: $m' \leftarrow [r]^{\eta,\tau}_{\Pi}$ and $n' = 0^{|m'|}$, and adds m' to vector $\vec{M}_i$ and n' to vector $\vec{N}_i$. Note that in order for $\mathcal{B}$ to evaluate m', it does not need to know the corresponding private key values of any of the keys in R, because their private keys do not appear as a message in e. After performing this operation for all elements in $undec_e(privkeys(e))$, it finally submits the two vectors $\vec{M} = (\vec{M}_1, \ldots, \vec{M}_{|R|})$ and $\vec{N} = (\vec{N}_1, \ldots, \vec{N}_{|R|})$. When provided with the challenge ciphertext vector $\vec{C}$, it uses $\vec{C}$ to produce a computational value E for e_1, and gives E to $\mathcal{A}^{pubkeys(e),\emptyset}$ and outputs whatever $\mathcal{A}^{pubkeys(e),\emptyset}(E)$ outputs. Before considering how $\mathcal{B}$ responds to $\mathcal{A}^{pubkeys(e),\emptyset}$'s oracle calls, note that if the challenge ciphertext $\vec{C}$ is an encryption of $\vec{M}$, then $\mathcal{A}^{pubkeys(e),\emptyset}$ is given a sample from $[e_1]^{\eta,\tau}_{\Pi}$, and if $\vec{C}$ is an encryption of $\vec{N}$ then $\mathcal{A}^{pubkeys(e),\emptyset}$ is given a sample from $[e_2]^{\eta,\tau}_{\Pi}$. So if $\mathcal{A}^{pubkeys(e)_1,\emptyset}$ is able to distinguish between $[e_1]^{\eta}_{\Pi}$ and $[e_2]^{\eta}_{\Pi}$ with non-negligible probability, then $\mathcal{B}$ breaks the CCA-2 secrecy assumption of Π. We now just need to show how $\mathcal{B}$ answers $\mathcal{A}^{pubkeys(e),\emptyset}$'s oracle queries. Suppose (σ, pk) is a query made by $\mathcal{A}^{pubkeys(e),\emptyset}$, $\mathcal{B}$ deals with the query as follow: It first checks whether $pk = \tau(k)$ for $k \in pubkeys(e)$, or not (note that $\mathcal{B}$ has the value of $\tau(k)$ for all $k \in pubkeys(e)$). If not, it means that (σ, pk) is an invalid query and so $\mathcal{B}$ returns $\perp$. Otherwise, noting that $FIX(\mathcal{F}_{e_1}) = FIX(\mathcal{F}_{e_2}) = T$, $\mathcal{B}$ checks whether $\langle \sigma, pk, \text{"ciphertext"} \rangle \in Vis_E(priv\_known)$, or not (note that $\mathcal{B}$ is able to do this because it has the private key values of all keys in $priv\_known$). If $\langle \sigma, pk, \text{"ciphertext"} \rangle \in Vis_E(priv\_known)$, it returns $\perp$ (because

in this case the query is invalid again), otherwise there are two cases: either $pk \in priv\_known$ in which case $\mathcal{B}$ has the private key value of pk and can decrypt σ, or $pk \in priv\_unknown$ in which case $\mathcal{B}$ asks its CCA-2 decryption oracle to decrypt σ. The fact that $\langle \sigma, pk, \text{"ciphertext"} \rangle \notin Vis_E(priv\_known)$ ensures that σ is a valid query and will be decrypted by the CCA-2 decryption oracle. □

Theorem 1 implies that for all $e \in Pat$, we have $[e]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),privkeys(e)} [p(e, privkeys(e))]_\Pi^{\eta,\tau}$. However, for our soundness results, we need to prove that in the above theorem if $T \subset privkeys(e)$, it still holds that $[e]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),T} [p(e, privkeys(e))]_\Pi^{\eta,\tau}$. Regardless of how it may appear at first, the proof of this fact turns out to be very involved. We present the proof via a sequence of intermediate lemmas.

Lemma 3. *Suppose* $\Pi = (Gen, Enc, Dec)$ *is CCA-2 secure,* $e \in Pat$*, and* $T \subset privkeys(e)$*. Then*

$$[e\ T\ T^{-1}]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),\emptyset} [p(e, privkeys(e))\ T\ T^{-1}]_\Pi^{\eta,\tau}$$

Proof. Let $T_1 = privkeys(e)$. In theorem 1 we showed that

$$[e\ T_1\ T_1^{-1}]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),\emptyset} [p(e, T_1)\ T_1\ T_1^{-1}]_\Pi^{\eta,\tau}$$

Suppose for the sake of contradiction there exists an adversary $\mathcal{A}^{pubkeys(e),\emptyset}$ which can distinguish between $[e\ T\ T^{-1}]_\Pi^{\eta,\tau}$ and $[p(e, T_1)\ T\ T^{-1}]_\Pi^{\eta,\tau}$ with non-negligible probability $q(\eta)$. We show that we can construct an adversary $\mathcal{B}^{pubkeys(e),\emptyset}$ to distinguish between $[e\ T_1\ T_1^{-1}]_\Pi^{\eta,\tau}$ and $[p(e, T_1)\ T_1\ T_1^{-1}]_\Pi^{\eta,\tau}$ with probability at least $q(\eta) - negl(\eta)$, contradicting theorem 1. The intuition behind $\mathcal{B}^{pubkeys(e),\emptyset}$ is clear; on input $r\mathcal{T}_1\mathcal{T}_1^{-1}$, where r is a sample drawn from either $[e]_\Pi^{\eta,\tau}$ or $[p(e, T_1)]_\Pi^{\eta,\tau}$, $\mathcal{T}_1 = \tau(T_1)$, and $\mathcal{T}_1^{-1} = \tau(T_1^{-1})$, it extracts the subset $\mathcal{T}$ of $\mathcal{T}_1$ which contains the computational values of T, and simulates $\mathcal{A}^{pubkeys(e),\emptyset}$ on $r\mathcal{T}\mathcal{T}^{-1}$, and tries to answer $\mathcal{A}$'s queries using its own private keys (i.e. set $\mathcal{T}_1$) or its own oracle. According to definition 6, the granted oracles to $\mathcal{A}$ and $\mathcal{B}$ would be $O_{r\mathcal{T}\mathcal{T}^{-1}}^{pubkeys(e),\emptyset}$ and $O_{r\mathcal{T}_1\mathcal{T}_1^{-1}}^{pubkeys(e),\emptyset}$, respectively. If we can show that all valid oracle calls that $\mathcal{A}$ may issue and can be answered by $O_{r\mathcal{T}\mathcal{T}^{-1}}^{pubkeys(e),\emptyset}$ can also be answered either by $\mathcal{B}$'s private keys (i.e. set $\mathcal{T}_1$) or by $O_{r\mathcal{T}_1\mathcal{T}_1^{-1}}^{pubkeys(e),\emptyset}$ (except with negligible probability), then we are done. So suppose (c, pk) is a query made by $\mathcal{A}$; now assuming that r is the computational value of e_i for unknown $i \in \{0, 1\}$, where $e_0 = e$, $e_1 = p(e, T_1)$, we have:

- (c, pk) is valid to $O^{pubkeys(e),\emptyset}_{r\mathcal{T}\mathcal{T}^{-1}} \Leftrightarrow \langle c, pk, \text{cipher}\rangle \notin Vis_{r\mathcal{T}\mathcal{T}^{-1}}(\tau(FIX^{-1}(\mathcal{F}_{e_iTT^{-1}}))) \Leftrightarrow \langle c, pk, \text{cipher}\rangle \notin Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$.
- (c, pk) is valid to $O^{pubkeys(e),\emptyset}_{r\mathcal{T}_1\mathcal{T}_1^{-1}} \Leftrightarrow \langle c, pk, \text{cipher}\rangle \notin Vis_{r\mathcal{T}_1\mathcal{T}_1^{-1}}(\tau(FIX^{-1}(\mathcal{F}_{e_iT_1T_1^{-1}}))) \Leftrightarrow \langle c, pk, \text{cipher}\rangle \notin Vis_r(\mathcal{T}_1^{-1}))$.

Thus, in order for $\mathcal{B}$ to determine whether (c, pk) is a valid query or not, it should check if $\langle c, pk, \text{cipher}\rangle \notin Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$. However $\mathcal{B}$ does not know whether $i = 0$ or $i = 1$ (i.e. $\mathcal{B}$ does not know whether r is the image of $e_0 = e$ or $e_1 = p(e, T_1)$), but this is not a problem since we can easily see that $FIX(\mathcal{F}_{e_0T}) = FIX(\mathcal{F}_{e_1T})$. Also, $FIX(\mathcal{F}_{e_iT}) \subseteq T_1$, which means that $\tau(FIX^{-1}(\mathcal{F}_{e_iT})) \subset \mathcal{T}_1^{-1}$; so $\mathcal{B}$ is able to determine whether $\langle c, pk, \text{cipher}\rangle \in Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$, or not. If $\langle c, pk, \text{cipher}\rangle \in Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$, then $\mathcal{B}$ returns $\perp$. Otherwise we have two cases:

1. $pk \in \mathcal{T}_1^{-1}$; In this case, $\mathcal{B}$ has the private key value of pk and can decrypt c.
2. $pk \notin \mathcal{T}_1^{-1}$: in this case, $\mathcal{B}$ queries (c, pk) from its own oracle $O^{pubkeys(e),\emptyset}_{r\mathcal{T}_1\mathcal{T}_1^{-1}}$. If the oracle decrypts for $\mathcal{B}$ (i.e. it does not return $\perp$), $\mathcal{B}$ gives the result to $\mathcal{A}$. Otherwise, there are two cases: either $\langle c, pk, \text{cipher}\rangle \in Vis_r(\mathcal{T}_1^{-1})$, or $pk \notin \tau(pubkeys(e))$. $\mathcal{B}$ checks if the former is the case (note that $\mathcal{B}$ is able to check $\langle c, pk, \text{cipher}\rangle \in^? Vis_r(\mathcal{T}_1^{-1})$, because $\mathcal{T}_1$ and $\mathcal{T}_1^{-1}$ are given to $\mathcal{B}$). If it is not, it should be the case that $pk \notin pubkeys(e)$, so $\mathcal{B}$ again returns $\perp$.

Thus, there is only one case that $\mathcal{B}$ is not able to decrypt: namely, $pk \notin \mathcal{T}_1^{-1}$, $\langle c, pk, \text{cipher}\rangle \notin Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$, and $\langle c, pk, \text{cipher}\rangle \in Vis_r(\mathcal{T}_1^{-1})$. Since $\langle c, pk, \text{cipher}\rangle \in Vis_r(\mathcal{T}_1^{-1})$ and $pk \notin \mathcal{T}_1^{-1}$, it implies that $\langle c, pk, \text{cipher}\rangle$ is an encoding of $\{m\}_k$ in r and $\{m\}_k \in undec_{e_i}(T_1)$. Also the fact that $\langle c, pk, \text{cipher}\rangle \notin Vis_r(\tau(FIX^{-1}(\mathcal{F}_{e_iT})))$ implies that $\{m\}_k \sqsubseteq \{m'\}_{k'}$ for $k' \notin FIX^{-1}(\mathcal{F}_{e_iT})$. Define:

$$Q = \{\{m\}_k | k^{-1} \notin privkeys(e_iT) \;\&\; \{m\}_k \sqsubseteq \{m'\}_{k'} \sqsubseteq e_i \text{ for } k' \notin FIX^{-1}(\mathcal{F}_{e_iT})\}$$

It turns out the set of oracle queries that $\mathcal{A}^{pubkeys(e),\emptyset}$ on input $r\mathcal{T}\mathcal{T}^{-1}$ is permitted to ask but cannot be answered by $\mathcal{B}$ is a subset of $r\mathcal{T}\mathcal{T}^{-1}{}_{[Q]}$. If we can show that the probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ asks such a query is negligible, then we are done. Namely, letting $H \leftarrow [e_iTT^{-1}]^{\eta,\tau}_{\Pi}$, we just need to prove that the probability that $\mathcal{A}^{pubkeys(e),\emptyset}(H)$ ever asks a query in $H_{[Q]}$ is negligible. Using the same technique presented in remark 1, it can be seen that this is equivalent to saying that the probability

that $\mathcal{A}^{pubkeys(e_i),\emptyset}(H)$ ever asks a query in $H_{[Q]}$ is negligible (because for every query (σ, pk), if $pk = \tau(k)$ for $k \in pubkeys(e) - pubkeys(e_i)$, k is either in the set T^{-1}, in which case $\mathcal{A}$ has the value of pk^{-1}, or $k \notin T^{-1}$ in which case the value of pk would be completely independent of H and the probability of such an oracle call is negligible). So suppose the probability that $\mathcal{A}^{pubkeys(e_i),\emptyset}(H)$ asks a query in $H_{[Q]}$ is non-negligible; since Q is finite, there exists $\{m\}_k \in Q$, an infinite set $\mathcal{N}$, and a constant c such that for all $\eta \in \mathcal{N}$, we have:

$$\Pr[\mathcal{A}^{e,\emptyset}(H) \text{ asks a query } (c, pk) \text{ for } \langle c, pk, \text{cipher}\rangle = H_{[\{m\}_k]}] \geq \eta^{-c}$$

We now show that this leads to the construction of an adversary $\mathcal{C}^{pubkeys(e_i),\emptyset}$ which on input H outputs $H_{[\{m\}_k]}$, rather than asking it as a query. The idea of $\mathcal{C}^{pubkeys(e_i),\emptyset}$ is clear: assuming $q(.)$ is a polynomial which upper-bounds the number of queries made by $\mathcal{A}^{pubkeys(e_i),\emptyset}$, $\mathcal{C}^{pubkeys(e),\emptyset}$ simulates $\mathcal{A}^{pubkeys(e_i),\emptyset}$ on its input and outputs the rth query made by $\mathcal{A}^{pubkeys(e_i),\emptyset}$, where $r \leftarrow^R \{1, ..., q(\eta)\}$. Obviously for all $\eta \in \mathcal{N}$, $\mathcal{C}^{pubkeys(e_i),\emptyset}$ outputs $H_{\{m\}_k}$ with probability at least $\frac{1}{\eta^c q(\eta)}$, which means that $\mathcal{C}^{pubkeys(e_i),\emptyset}$ succeeds with non-negligible probability. With some trivial modification on $\mathcal{C}$, we can see that $\mathcal{C}^{pubkeys(e_i T T^{-1}),\emptyset}$ also succeeds with non-negligible probability (this is very trivial, because $\mathcal{C}$ on H does not need to ask its oracle to decrypt with respect to keys in $\tau(T^{-1})$ as their private key values are provided as part of H). However in the following lemma, we show that this is indeed a contradiction, and this will complete the proof of lemma 3:

Lemma 4. *Suppose $e \in Pat$, $\{m\}_k$ is a subterm of e such that $k^{-1} \notin privkeys(e)$, and $\{m\}_k \sqsubseteq \{m_1\}_{k_1} \sqsubseteq e$ for $k_1^{-1} \notin FIX(F_e)$. Then for every adversary $\mathcal{A}$ it holds that:*

$$pr_{E \leftarrow [e]_\Pi^{\eta,\tau}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{\{m\}_k}] = negl(\eta)$$

We give the proof of the above lemma via a sequence of intermediate lemmas (lemmas 5, 6, 7). For convenience, we may sometimes write $f(\eta) \neq negl(\eta)$ to mean the function f is non-negligible. For $e \in Pat$, let's define:

$$W_e = \{\, \{m\}_k \mid \{m\}_k \in undec_e(privkeys(e)) \;\&\; \{m\}_k \sqsubseteq \{m'\}_{k'} \sqsubseteq e \text{ for } k'^{-1} \notin FIX(F_e)\}$$

Lemma 5. *Let $e \in Pat$, and $\{m\}_k \in W_e$. If there exists an adversary $\mathcal{A}$ such that $pr_{E \leftarrow [e]_\Pi^{\eta,\tau}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{[\{m\}_k]}] \neq negl(\eta)$, then there exists $\{m'\}_{k'} \in W_e$, and an adversary $\mathcal{B}$ such that the probability that $\mathcal{B}^{pubkeys(e),\emptyset}$ when given $E \leftarrow [e]_\Pi^{\eta,\tau}$ outputs*

$E_{[\{m'\}_{k'}]}$ and does not ask any query (σ, pk) for $\langle \sigma, pk, cipher\rangle \in E_{[W_e]}$ is non-negligible.

Proof. Giving $E \leftarrow [e]^{\eta,\tau}_{\Pi}$ to an adversary as input, we call a query (σ, pk) made by the adversary "unfavorable" if $\langle \sigma, pk, \text{cipher}\rangle \in E_{[W_e]}$. Suppose $pr_{E\leftarrow[e]^{\eta,\tau}_{\Pi}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{[\{m\}_k]}] \neq negl(\eta)$. Let $p_1(\eta)$ be the probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ during its computation on E asks an "unfavorable" query. If $p_1(\eta)$ is negligible then we are done. Thus, suppose $p_1(\eta)$ is not negligible. Since W_e is finite and $p_1(\eta)$ is non-negligible, there exists $\{m'\}_{k'} \in W_e$ and an infinite set $\mathcal{N}$ such that for all $\eta \in \mathcal{N}$, the very first "unfavorable' query made by $\mathcal{A}$ is the computational image of $\{m'\}_{k'}$ with probability greater than η^{-c} for some constant c. Now let $q(.)$ be a polynomial which upper-bounds the number of queries made by $\mathcal{A}^{pubkeys(e),\emptyset}$. Consider the adversary $\mathcal{B}^{pubkeys(e),\emptyset}$ which starts by selecting $r \leftarrow \{1, ..., q(\eta)\}$ uniformly at random, and then simulates $\mathcal{A}^{pubkeys(e),\emptyset}(E)$ and outputs the rth query made by $\mathcal{A}$. Now for all $\eta \in \mathcal{N}$ the probability that $\mathcal{B}$ hits and outputs the very first "unfavorable" query is at least $\frac{\eta^{-c}}{q(\eta)} \geq \eta^{-c'}$, for some constant c'. This completes the proof.

□

Lemma 6. *Suppose $e \in Pat$, $e_1 = p(e, privkeys(e))$, and $\{m'\}_{k'} \in W_e$. If the probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on $E \leftarrow [e]^{\eta,\tau}_{\Pi}$ outputs $E_{[\{m'\}_{k'}]}$ and does not ask any query (σ, pk) for $\langle \sigma, pk, cipher\rangle \in E_{[W_e]}$ is non-negligible, then the probability that $\mathcal{A}^{pubkeys(e_1),\emptyset}$ on input $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ computes $E_{1[\{struct(m')\}_{k'}]}$ is also non-negligible.*

Proof. Let $W'(e) = \{\ \{struct(m)\}_k \mid \{m\}_k \in W_e\ \}$. Note that W'_e is the transformation of W_e in e_1 (i.e. subexpressions W_e of e are transformed into W'_e in e_1). Also define:

- $q_1(\eta)$: Probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on $E \leftarrow [e]^{\eta,\tau}_{\Pi}$ outputs $E_{[\{m'\}_{k'}]}$ and does not ask any query (σ, pk) for $\langle \sigma, pk, \text{cipher}\rangle \in E_{[W_e]}$
- $q'_1(\eta)$: Probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ outputs $E_{1[\{struct(m')\}_{k'}]}$ and does not ask any query (σ, pk) for $\langle \sigma, pk, \text{cipher}\rangle \in E_{[W'_e]}$
- $q'_2(\eta)$: Probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on input $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ outputs $E_{1[\{struct(m')\}_{k'}]}$
- $q'_3(\eta)$: Probability that $\mathcal{A}^{pubkeys(e_1),\emptyset}$ on input $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ outputs $E_{1[\{struct(m')\}_{k'}]}$

First of all note that $|q'_3(\eta) - q'_2(\eta)|$ is negligible. This is because of the fact that the probability that $\mathcal{A}$ when given $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ issues a query (c, pk) such that $pk = \tau(k)$ for $k \in pubkeys(e) - pubkeys(e_1)$ is negligible. Thus, we just need to prove

that $q_2'(\eta)$ is non-negligible. Note that $q_1'(\eta) < q_2'(\eta)$, so it suffices to prove that $q_1'(\eta)$ is non-negligible. To do this, we prove that $|q_1(\eta) - q_1'(\eta)|$ is negligible. In particular, we prove if the above difference is not negligible, we can construct an adversary $\mathcal{B}$ to break the CCA-2 secrecy of Π. The adversary $\mathcal{B}$ works exactly like the adversary $\mathcal{B}$ introduced in the proof of theorem 1: for every private key $k^{-1} \in privkeys(e)$, it runs the key generation algorithm to obtain a pair of key values $(\tau(k), \tau(k^{-1}))$ for (k, k^{-1}), and adds $\tau(k)$ to the set *priv_known*. Letting $R = pubkeys(e) - privkeys^{-1}(e) = \{k_1, \ldots, k_{|R|}\}$, we assume that the multiple message indistinguishability experiment that $\mathcal{B}$ is involved in is run under $|R|$ randomly chosen public keys $pk_1, \ldots, pk_{|R|}$. Let $priv\_unknown = \{pk_1, ..., pk_{|R|}\}$ and $\tau(k_i) = pk_i$. As the proof of theorem 1, let $\vec{M}$ and $\vec{N}$ be the two vectors that $\mathcal{B}$ submits for encryption, and $\vec{C}$ be the challenge ciphertext vector it receives (for details on how $\vec{M}$ and $\vec{N}$ are constructed, see the proof of theorem 1). Also suppose $\mu \in C$ is the ciphertext which is the encryption of either $\{m'\}_{k'}$ or $\{struct(m')\}_{k'}$, depending on whether $\vec{C}$ was an encryption of $\vec{M}$ or $\vec{N}$. Now based on $\vec{C}$, $\mathcal{B}$ randomly constructs the bit-string H which is the computational image of either e or e_1, and gives H to $\mathcal{A}^{pubkeys(e),\emptyset}$. Denoting by $\vec{C}'$ the subset of $\vec{C}$ which corresponds to computational images of either W_e or W_e', if $\mathcal{A}^{pubkeys(e),\emptyset}(H)$ during its computation asks a query (σ, pk) for $\langle \sigma, pk, \text{cipher} \rangle \in \vec{C}'$, $\mathcal{B}$ stops and returns $i \leftarrow^R \{0,1\}$. If $\mathcal{A}$ does not ask such a query and finally outputs μ, $\mathcal{B}$ returns 1. Otherwise, $\mathcal{B}$ returns $i \leftarrow^R \{0,1\}$. For oracle calls that $\mathcal{B}$ does not stop, it can answer them appropriately as the proof of theorem 1. It is now easy to see that:

$\mathcal{B}$'s probability of success $= \frac{1}{2}(|q_1(\eta) + \frac{1-q_1(\eta)}{2} + \frac{1-q_1'(\eta)}{2}|) = \frac{1}{2} + \frac{|q_1(\eta)-q_1'(\eta)|}{4}$

Thus it is proved that $q_1(\eta) - q_1'(\eta)$ must be negligible (because otherwise $\mathcal{B}$ breaks the CCA-2 secrecy of Π), and the proof is complete.

□

Lemma 7. *Suppose $e \in Pat$, $\{m\}_k$ is a subterm of e such that $k^{-1} \notin privkeys(e)$, and $\{m\}_k \sqsubseteq \{m_1\}_{k_1} \sqsubseteq e$ for $k_1^{-1} \notin FIX(F_e)$. If there exists an adversary $\mathcal{A}$ for which $pr_{E \leftarrow [e]_\Pi^{\eta,\tau}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{[\{m\}_k]}]$ is non-negligible, then there exists an adversary $\mathcal{B}$, and $\{m'\}_{k'} \in W_e$ such that $pr_{E \leftarrow [e]_\Pi^{\eta,\tau}}[\mathcal{B}^{pubkeys(e),\emptyset}(E) = E_{[\{m'\}_{k'}]}]$ is also non-negligible.*

Proof. Suppose $pr_{E \leftarrow [e]_\Pi^{\eta,\tau}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{[\{m\}_k]}] = q(\eta) \neq negl(\eta)$. First note that, if $\{m\}_k \in undec_e(privkeys(e))$ then we are done; otherwise there are two cases:

Case 1: The probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on input $E \leftarrow [e]_\Pi^{\eta,\tau}$ asks a query (σ, pk) for $\langle \sigma, pk, \text{cipher} \rangle \in E_{[W_e]}$ is non-negligible. Then, as we have proved before, there

exists an adversary $\mathcal{B}$ and $\{m_1\}_{k_1} \in W_e$ such that $pr_{E \leftarrow [e]^{\eta}_{\Pi}}[\mathcal{B}^{pubkeys(e),\emptyset}(E) = E_{[\{m_1\}_{k_1}]}]$ is also non-negligible.

Case 2: The probability that $\mathcal{A}^{pubkeys(e),\emptyset}$ on input $E \leftarrow [e]^{\eta,\tau}_{\Pi}$ asks a query (σ, pk) for $\langle \sigma, pk, \text{cipher} \rangle \in E_{[W_e]}$ is negligible. We prove that, under the assumption $\{m\}_k \notin undec_e(privkeys(e))$ we made, this leads to a contradiction. First note that since $\{m\}_k \notin undec_e(privkeys(e))$, there exists $\{m'\}_{k'}$ such that $\{m\}_k \sqsubseteq \{m'\}_{k'}$ and $\{m'\}_{k'} \in undec_e(privkeys(e))$. Now, we can construct an adversary $\mathcal{B}$ to break the CCA-2 secrecy of Π as follow: let CCA-2 indistinguishability experiment run under a single key pk, and think of pk as the computational value of k' (i.e. $\tau(k') = pk$. Remember that k'^{-1} does not occur in e). For all other key symbols k'' that occur in e, $\mathcal{B}$ assigns $\tau(k'')$ by running the key generation algorithm. Now $\mathcal{B}$ draws two random samples P_1, $P_2 \leftarrow [\{m\}_k]^{\eta,\tau}_{\Pi}$, and using P_1 and P_2 as the values for $\{m\}_k$, it evaluates two random samples M_1 and M_2 from $[m']^{\eta}_{\Pi}$, and submits them for encryption under pk (note that we have $M_{i[\{m\}_k]} = P_i$ for $i \in \{1, 2\}$). When provided with the challenge ciphertext C, $\mathcal{B}$ uses C to evaluate a sample from $[e]^{\eta,\tau}_{\Pi}$, and passes on the result to $\mathcal{A}^{pubkeys(e),\emptyset}$. Finally, if $\mathcal{A}^{pubkeys(e),\emptyset}$ outputs P_i for $i \in \{1, 2\}$, $\mathcal{B}$ returns i; otherwise, $\mathcal{B}$ returns $i \leftarrow^R \{1, 2\}$. Now it can be easily seen if $pr_{E \leftarrow [e]^{\eta}_{\Pi}}[\mathcal{A}^{pubkeys(e),\emptyset}(E) = E_{[\{m\}_k]}] = q(\eta)$, the probability that $\mathcal{B}$ succeeds in the CCA-2 experiment is at least $q(\eta) + \frac{1-q(\eta)}{2} = \frac{1}{2} + \frac{q(\eta)}{2}$. Also note that $\mathcal{B}$ is able to detect the validity and answer the oracle queries made by $\mathcal{A}^{pubkeys(e),\emptyset}$. In particular, if (σ, pk_1) is a valid query and $pk_1 \neq pk$, $\mathcal{B}$ has the private key of pk_1 and so is able to decrypt σ. In case that $pk_1 = pk$, it will ask the query from its CCA-2 decryption oracle which will decrypt if $\sigma \neq C$. But can it be the case that $\sigma = C$? Depending on whether $\{m'\}_{k'}$ is in W_e or not we have two cases: either $\{m'\}_{k'} \notin W_e$ which implies that $\{m'\}_{k'}$ is co-inductively derivable from e (because we already know that $\{m'\}_{k'} \in undec_e(privkeys(e))$), and consequently condition 2 of definition 6 guarantees that $\sigma \neq C$ and so $\mathcal{B}$ can ask its own CCA-2 decryption oracle to decrypt σ, or $\{m'\}_{k'} \in W_e$, in which case the fact that $\mathcal{A}^{pubkeys(e),\emptyset}$ on E asks a query in $E_{[W_e]}$ with negligible probability implies that the probability that $\sigma = C$ is negligible. So the probability that $\mathcal{B}$ is not able to answer a requested query of $\mathcal{A}^{pubkeys(e),\emptyset}$ is negligible, and this completes the proof. □

Proof of lemma 4 First of all note that according to lemma 7, it implies that there exists $\{m_0\}_{k_0} \in W_e$, and an adversary $\mathcal{A}_0$ such that $pr_{E \leftarrow [e]^{\eta,\tau}_{\Pi}}[\mathcal{A}_0^{pubkeys(e),\emptyset}(E) = E_{[\{m_0\}_{k_0}]}] \neq negl(\eta)$. Define: $e_0 = e$ and $e_i = p(e_{i-1}, privkeys(e_{i-1}))$ for $i \geq 1$. Let-

ting $r = |privkeys(e)|$, we have $pat_C(e) = e_r$. We prove that the existence of an adversary $\mathcal{A}_i$, and $\{m_i\}_{k_i} \in W_{e_i}$ such that $pr_{E_i \leftarrow [e_i]^{\eta,\tau}_{\Pi}}[\mathcal{A}_i^{pubkeys(e_i),\emptyset}(E_i) = E_{i[\{m_i\}_{k_i}]}] \neq negl(\eta)$ results in the existence of an adversary $\mathcal{A}_{i+1}$, and $\{m_{i+1}\}_{k_{i+1}} \in W_{e_{i+1}}$ such that $pr_{E_{i+1} \leftarrow [e_{i+1}]^{\eta,\tau}_{\Pi}}[\mathcal{A}_{i+1}^{pubkeys(e_{i+1}),\emptyset}(E_{i+1}) = E_{i+1[\{m_{i+1}\}_{k_{i+1}}]}] \neq negl(\eta)$. In particular, this implies that there exists $\{m_r\}_{k_r} \in W_{pat_C(e)}$ and an adversary $\mathcal{A}_r$ such that $pr_{E_r \leftarrow [e_r]^{\eta,\tau}_{\Pi}}[\mathcal{A}^{pubkeys(e_r),\emptyset}(E_r) = E_{r[\{m_r\}_{k_r}]}] \neq negl(\eta)$, but this is a contradiction since $W_{pat_C(e)} = \emptyset$ and so such an $\{m_r\}_{k_r}$ does not exist. So we prove the above claim; suppose $\{m_i\}_{k_i} \in W_{e_i}$ and $pr_{E_i \leftarrow [e_i]^{\eta,\tau}_{\Pi}}[\mathcal{A}_i^{pubkeys(e_i),\emptyset}(E_i) = E_{i[\{m_i\}_{k_i}]}] \neq negl(\eta)$. Then lemmas 5 and 6 together yields that there exists an adversary $\mathcal{B}$ for which $pr_{E_{i+1} \leftarrow [e_{i+1}]^{\eta,\tau}_{\Pi}}[\mathcal{B}^{pubkeys(e_{i+1}),\emptyset}(E_{i+1}) = E_{i+1[\{m'\}_{k'}]}] \neq negl(\eta)$, where $\{m'\}_{k'} \sqsubseteq \{m''\}_{k''} \sqsubseteq e_{i+1}$, $k'^{-1} \notin privkeys(e_{i+1})$, and $k'' \notin FIX(F_{e_{i+1}})$. It now follows from lemma 7 that there exists $\{m_{i+1}\}_{k_{i+1}} \in W_{e_{i+1}}$, and an adversary $\mathcal{A}_{i+1}$ such that $pr_{E_{i+1} \leftarrow [e_{i+1}]^{\eta,\tau}_{\Pi}}[\mathcal{A}_{i+1}^{pubkeys(e_{i+1}),\emptyset}(E_{i+1}) = E_{i+1[\{m_{i+1}\}_{k_{i+1}}]}] \neq negl(\eta)$. This completes the proof.

□

According to what we pointed out before, lemma 3 implies that:

$$[e]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e),T} [p(e, privkeys(e))]^{\eta,\tau}_{\Pi} \tag{4.1.1}$$

Theorem 2. *Suppose* Π *is CCA-2 secure,* $e \in Pat$*, and* $T \subseteq FIX(F_e)$*. Then:*

$$[e]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e),T^{-1}} [pat_C(e)]^{\eta,\tau}_{\Pi}$$

Proof. It suffices to prove the theorem for the case that $pubkeys(e) = pubkeys(pat_C(e))$. In order to see this, suppose we have proved it for this especial case and we want to prove it for e that $pubkeys(e) \neq pubkeys(pat_C(e))$. Let $e' = e\ T_1$, where $T_1 = pubkeys(e) - pubkeys(pat_C(e))$. Noting that $pat_C(e') = pat_C(e)\ T_1$ and $pubkeys(e) = pubkeys(eT_1)$, we have $[e\ T_1]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(eT_1),T^{-1}} [pat_C(e)T_1]^{\eta,\tau}_{\Pi}$, and this immediately implies that it should hold that $[e]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e),T^{-1}} [pat_C(e)]^{\eta,\tau}_{\Pi}$.

So suppose $pubkeys(e) = pubkeys(pat_C(e))$. Let $e_0 = e$, and $e_i = p(e_{i-1}, privkeys(e_{i-1}))$, for $i \geq 1$. Assuming $n = |privkeys(e)|$, we have $e_n = pat_C(e)$. Also for all i we have $pubkeys(e_i) = pubkeys(e)$ (because in general we have $pubkeys(pat_C(e)) \subseteq pubkeys(e_i) \subseteq pubkeys(e)$). Now according to equation 4.1.1 we have:

$$[e]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e),T^{-1}} [e_1]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e_1),T^{-1}} ...[e_i]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e_i),T^{-1}} [e_{i+1}]^{\eta,\tau}_{\Pi} \approx_C^{pubkeys(e_{i+1}),T^{-1}} ...[e_n]^{\eta,\tau}_{\Pi}$$

Now since for every i and j we have $pubkeys(e_i) = pubkeys(e_j)$, the two relations $\approx_C^{pubkeys(e_i),T^{-1}}$ and $\approx_C^{pubkeys(e_j),T^{-1}}$ coincide, and so the transitivity holds and we have $[e]_\Pi^{\eta,\tau} \approx_C^{pubkeys(e),T^{-1}} [pat_C(e)]_\Pi^{\eta,\tau}$. This completes the proof. □

Corollary 1. *If* Π *is CCA-2 secure, then it provides co-inductive strong indistinguishability in the presence of key-cycles. That is, for* $m \in Pat$, *and* $T \subseteq \mathcal{K}_{pub}$, *it holds that:*

$$[m]_\Pi^\eta \approx_C^{pubkeys(m),T} [pat_C(m, T^{-1})]_\Pi^\eta$$

Proof. The proof is very easy; according to theorem 2, we have $[mT^{-1}]_\Pi^\eta \approx_C^{pubkeys(m),T} [pat_C(m, T^{-1})T^{-1}]_\Pi^\eta$, and this implies $[m]_\Pi^\eta \approx^{pubkeys(m),T} [pat_C(m, T^{-1})]_\Pi^\eta$.

□

Corollary 2. *(Computational soundness) Suppose* $m, n \in Exp$, Π *is CCA-2 secure, and* $T_1 = pubkeys(m) \cup pubkeys(n)$. *Then:*

$$pat_C(m, T) = pat_C(n, T) \rightarrow [m]_\Pi^{\eta,\tau} \approx_C^{T_1,T^{-1}} [n]_\Pi^{\eta,\tau}$$

Proof. By corollary 1, $[mT_1]_\Pi^{\eta,\tau} \approx_C^{T_1,T^{-1}} [pat_C(m, T)T_1]_\Pi^{\eta,\tau}$, and $[nT_1]_\Pi^\eta \approx_C^{T_1,T^{-1}} [pat_C(n, T)T_1]_\Pi^{\eta,\tau}$. Now since $[pat_C(n, T)T_1]_\Pi^{\eta,\tau} = [pat_C(m, T)T_1]_\Pi^{\eta,\tau}$, it holds $[mT_1]_\Pi^{\eta,\tau} \approx_C^{T_1,T^{-1}} [nT_1]_\Pi^\eta$. It immediately follows that $[m]_\Pi^{\eta,\tau} \approx_C^{T_1,T^{-1}} [n]_\Pi^{\eta,\tau}$, and the proof is complete. □

Using the same approach taken for establishing the proof of corollary 1, one can prove a weaker result stating that if Π is IND-CPA secure, then it provides weak indistinguishability property (definition 8). In fact, the proof of this result is very analogous its private key version of [27]:

Theorem 3. *IND-CPA security provides weak indistinguishability property in the presence of key-cycles. That is, if* Π *is an IND-CPA secure encryption scheme, then it holds:*

$$e_1 \cong_C e_2 \Rightarrow [e_1]_\Pi^\eta \approx [e_2]_\Pi^\eta$$

In the next chapter, we show that IND-CCA1 secrecy, which is strictly stronger than IND-CPA secrecy, is still insufficient for providing strong indistinguishability even in the absence of key-cycles.

4.2 Non-malleability and indistinguishability

We will first show that co-inductive Dolev-Yao non-malleability is provided by CCA-2 secure encryption schemes in the presence of key-cycles, extending the result of [21]. That is, we show that if a computational adversary is given the instantiation of a formal expression under a CCA-2 secure encryption scheme, then it is not able to produce a possible interpretation of an expression outside the co-inductive closure set of the original expression.

4.2.1 Indistinguishability is strictly stronger than Non-Malleability

Theorem 4. *If an encryption scheme* Π *provides CCA-2 secrecy, it also provides co-inductive Dolev-Yao non-malleability in the presence of key-cycles (definition 10).*

Proof. The proof of this theorem is a consequence of Corollary 1 and the results of [21]. First of all, in [21] it is proved that inductive strong indistinguishability implies inductive Dolev-Yao non-malleability (theorem 18 in [21]). With exactly the same proof technique, it can be proved that in the presence of key-cycles, co-inductive strong indistinguishability implies co-inductive Dolev-Yao non-malleability (we very briefly give the idea of the proof below. We refer the reader to [21] for a full proof). Now having proved that CCA-2 secrecy provides co-inductive strong indistinguishability in the presence of key-cycles(corollary 1), the result follows.

Suppose Π is an encryption scheme for which there exists S, e and an adversary $\mathcal{A}$ which violates definition 10. Now note that since $e \notin Closure_C[S]$, there exists a path in the parse tree of e such that no element along the path is in $Closure_C[S]$, and moreover this path ends up in a leaf which is a private key symbol $k^{-1} \notin Closure_C(S)$. Given this k^{-1}, let $S' = (\hat{S}, b, \{b\}_k, k)$, where $b \in Block$ and, assuming $S = \{e_1, \ldots, e_l\}$, $\hat{S}$ denotes the single expression $(e_1, \ldots, e_l)$. Defining $T = pubkeys(e) - pubkeys(S')$, we can now construct an adversary $\mathcal{B}^{pubkeys(S'),T}$ from $\mathcal{A}$ to distinguish between $[S']_{\Pi}^{\eta,\tau}$ and $[pat_C(S', T^{-1})]_{\Pi}^{\eta,\tau}$, meaning that Π does not give co-inductive strong non-malleability. □

In the following theorem, by providing a counter-example, we show that although co-inductive strong indistinguishability implies co-inductive Dolev-Yao non-malleability, the converse of this is not true. The counter-example that we present is also applicable to the inductive setting, implying that in both settings, strong indistinguishability property is strictly stronger than the non-malleability property.

Theorem 5. *Co-inductive Dolev-Yao non-malleability does not imply co-inductive strong indistinguishability. That is, if there exists an scheme that provides co-inductive Dolev-Yao non-malleability, then there exists another scheme that provides co-inductive Dolev-Yao non-malleability but not co-inductive strong indistinguishability.*

Proof. Suppose $\Pi = (Gen, Enc, Dec)$ is an encryption scheme which provides co-inductive non-malleability. Also assume that Π provides co-inductive strong indistinguishability (because otherwise we are done). We construct $\Pi' = (Gen, Enc', Dec')$ from Π in such a way that Π' provides co-inductive non-malleability but not co-inductive strong indistinguishability. First of all, according to what we have assumed before, we have $|Enc_{pk}(x)| \geq |x|$ for all x's in the plaintext space of Enc_{pk}. Now for every pk and sk we define: $Enc'_{pk}(x) = Enc_{pk}(x)$, if $x \notin \{0,1\}$, and we leave $Enc'_{pk}(.)$ undefined for $x \in \{0,1\}$. Also we define $Dec'_{sk}(x) = Dec_{sk}(x)$ if $x \notin \{0,1\}$, and we define $Dec'_{sk}(0) = sk$. It is easy to see that Π' still provides co-inductive non-malleability. The reason for this is that for all security parameters η, $e \in Pat$, and $E \in supp[e]^{\eta,\tau}_{\Pi}$, it holds $|E| > 1$ (this is because of the fact that each bit-string is tagged with its type which enforces its length to be greater than one), and this implies that $[e]^{\eta,\tau}_{\Pi} = [e]^{\eta,\tau}_{\Pi'}$. However, Π' does not provide co-inductive strong indistinguishability; we can easily construct an adversary $\mathcal{A}^{pubkeys(e),\emptyset}$ to distinguish between $[\{k_1\}_k]^{\eta,\tau}_{\Pi'}$ and $[\{\circ\}_k]^{\eta,\tau}_{\Pi'}$. The idea is simple; upon receiving $E_1 = \langle E, pk, \text{"ciphertext"}\rangle$ which is either an image of $\{k_1\}_k$ or $\{\circ\}_k$, it asks its oracle to decrypt 0 with respect to pk (which is a valid query because $\langle 0, pk, \text{"ciphertext"}\rangle \notin Vis_{E_1}(\emptyset)$). It will receive the private key of pk this way, and, thus, will accomplish its distinguishing task. □

One might argue at this point that the strict strength of strong indistinguishability over non-malleability is due to the fact that the strong indistinguishability notion allows the presence of decryption oracles, while such a provision is not provided in our non-malleability notion. Thus, alternatively, one might consider strengthening our non-malleability notion by providing the computational adversary with a decryption oracle of similar type presented in definition 6 (we refer the reader to the conclusion chapter for further discussion on this matter). However, as we demonstrate in the next sub-chapter, although our non-malleability notion makes no use of decryption oracles, it is still too coarse to be satisfied by IND-CCA1 secure encryption (and consequently any weaker notion such as IND-CPA).

4.2.2 Non-Malleability Is Not Implied by Weaker Notions of Computational Security

Theorem 6. *IND-CCA1 secrecy does not provide co-inductive Dolev-Yao non-malleability. That is, if there exists an IND-CCA1 secure encryption scheme, then there exists an scheme that is IND-CCA1 secure but does not provides co-inductive Dolev-Yao non-malleability.*

Proof. Suppose $\Pi = (Gen, Enc, Dec)$ is IND-CCA1 secure. Let $d_\eta = |\langle sk, \text{"privkey"}\rangle|$ and $l_\eta = |\langle Enc_{pk}(\langle sk, \text{"privkey"}\rangle), pk, \text{"ciphertext"}\rangle|$ where $(pk, sk) \leftarrow Gen(1^\eta)$. Define

$$Enc'(m, pk) = \begin{cases} Enc_{pk}(m) \mid\mid \langle\ Enc_{pk}(x), pk, \text{"ciphertext"}\rangle & m = \langle 0, x, \text{"pair"}\ \rangle \text{ for} |x| = d_\eta \\ Enc_{pk}(m) \mid\mid \langle\ Enc_{pk}(0^{d_\eta}), pk, \text{"ciphertext"}\ \rangle & \text{otherwise} \end{cases}$$

For decryption, if $c = c_1||c_2$ for $|c_2| = l_\eta$, then $Dec'_{sk}(c) = Dec_{sk}(c_1)$. Then it is not hard to see that $\Pi' = (Gen, Enc', Dec')$ is IND-CCA1 secure. Now let $e = \{(0, k^{-1})\}_{k_1}$, $\langle E, pk, \text{"ciphertext"}\rangle \leftarrow [e]^{\eta,\tau}_{\Pi'}$, and E_1 be the bit-string containing the last l_η bits of E, we have $E_1 \in supp[\{k^{-1}\}_{k_1}]^{\eta,\tau}_{\Pi'}$ with probability one. However, $\{k^{-1}\}_{k_1} \notin Closure_C(e)$, and this completes the proof.

□

The above result has many powerful implications. First of all, note that the counter-example presented above is a key-acyclic expression, which implies that even in the absence of key-cycles, IND-CCA1 secrecy is insufficient for providing Dolev-Yao non-malleability. Moreover, the above counter-example can be slightly modified to give a constructive proof that IND-CCA1 does not imply co-inductive strong indistinguishability even in the absence of key-cycles: let Π and e be the same as above. We claim that the *PPT* adversary $\mathcal{A}^{pubkeys(e),\emptyset}$ which is working as follow can easily distinguish between $e = \{(0, k^{-1})\}_{k_1}$ and $pat_C(e) = \{(\square, \circ_p)\}_{k_1}$: upon receiving $E = \langle E_1, pk_1, \text{"ciphertext"}\rangle$ where $E \leftarrow [e]^{\eta,\tau}_\Pi$ or $E \leftarrow [Pat_C(e)]^{\eta,\tau}_\Pi$, it forms a new bit-string E_2 which is the last l_η bits of E_1. Note that $E_2 = \langle E_3, pk_1, \text{"ciphertext"}\rangle$ for some E_3. It now asks its oracle to decrypt E_3 with respect to pk_1, which is a valid query because $\langle E_3, pk_1, \text{"ciphertext"}\rangle \notin Vis_E(\emptyset)$, and, thence, will accomplish its distinguishing task. Note that the above counter-example is also applicable to the inductive setting, showing that in the absence of key-cycles, IND-CCA1 secrecy does not imply inductive strong indistinguishability.

Corollary 3. *In both the inductive and co-inductive settings we have:*

- *IND-CCA1 security does not provide Dolev-Yao non-malleability, even in the absence of key-cycles.*
- *IND-CCA1 security does not provide strong indistinguishability.*

Secondly, the above theorem implies that, in the co-inductive setting, weak indistinguishability does not imply Dolev-Yao non-malleability. To see this, recall that at the end of the previous chapter, we showed that IND-CCA1 secrecy implies weak indistinguishability. Now if Dolev-Yao non-malleability is implied by weak indistinguishability, then IND-CCA1 secrecy implies non-malleability, which is a contradiction.

Corollary 4. *Co-inductive weak indistinguishability does not imply co-inductive Dolev-Yao non-malleability.*

4.3 Indistinguishability-Computational Completeness

4.3.1 Background

Now that we have shown that the "formal equivalence" relation induced by the co-inductive pattern semantics is computationally sound. One natural question arisen at this point is that whether the co-inductive pattern semantics is complete with respect to the computational semantics, or not. In other words, if the associated probability ensembles associated with two expressions are computationally indistinguishable (strong or weak), is it true that the two expressions have the same pattern. This question has been investigated in some previous research efforts in the setting of private-key encryption [29, 22]. One feature, and still limitation, of these research works is that their defined pattern semantics treat all undecipherable parts of an expressions similarly. That is, their pattern semantics replace the underlying plaintext of any undecryptable ciphertext by $\Box$ (i.e. they define $p(\{e\}_k, T) = \Box$ if $k^{-1} \notin T$), not taking into account such considerations as the length or the structure of the plaintext. On the other hand, it turns out if one wants to relax these hindering restrictions in the pattern semantics definition, similar to our formal semantics, some

additional difficulty will arise when covering the completeness problem, which was absent in previous work. Let's elaborate more on why the completeness problem becomes more challenging in our framework. Consider as an example two expressions e_1 and e_2 which have different structures. Now with respect to our formal semantics, the two expressions $\{e_1\}_k$ and $\{e_2\}_k$ are inequivalent. As a result, in order to have completeness, the computational images of e_1 and e_2 should be of different length. In other words:

$$struct(e_1) \neq struct(e_2) \Rightarrow |[e_1]| \neq |[e_2]| \tag{4.3.1}$$

This is the very least condition that needs to be satisfied in order for our formal semantics to be computationally complete. As a result, we have to see under what conditions we can ensure that the computational interpretation of formal expressions of different structures results in bit-strings of different length. As it can be seen, the length of the computational image of a formal expression e depends both on the underlying encryption function and the computational pairing function. However, the encryption function is not selected by us, it depends on the encryption scheme that we study, and, thence, its length function could be different from one encryption scheme to another. As a result, we have to investigate if we can come up with an appropriate pairing function relative to which condition 4.3.1 holds, no matter what the encryption function is. Note that for many pairing functions the condition 4.3.1 is very unlikely to hold. For example, suppose e_1 and e_2 have different structures. Then it is likely that under an inappropriately-chosen pairing function, the two terms (e_1, e_2) and (e_2, e_1) be evaluated to equal-length bit-strings, breaking down the completeness of our formal semantics. This is the subject of the current work. In particular, we will prove that in the absence of key-cycles, if the computational pairing function is chosen properly, the co-inductive formal semantics is complete with respect to the computational semantics built upon any *length-revealing* encryption scheme (a property to be defined later). However, by providing a counter-example, we prove that the completeness result does not hold even with respect to IND-CPA secure encryption schemes in the presence of key-cycles (this is independent of the choice of the pairing function).

4.3.2 Presence of key-cycles

We show that in the presence of key-cycles, IND-CPA security does not provide completeness for our co-inductive formal semantics. That is, we prove that there exists

certain IND-CPA secure encryption schemes under which two expressions which are co-inductively inequivalent are mapped to computationally indistinguishable ensembles. We, however, need to make an assumption in our proof for incompleteness; we assume that there exists an IND-CPA secure encryption scheme Π which has the property that there exists only one matching private key for any public key. In other words, for all security parameters η, if it is the case that $(pk, sk_1) \in supp(Gen(\eta))$ and $(pk, sk_2) \in supp(Gen(\eta))$ then it should hold that $sk_1 = sk_2$. We call encryption schemes which satisfy this property *key-bijective*, and we show that under this assumption, completeness does not hold. Let's first prove a lemma that will be used in our theorem:

Lemma 8. *Suppose* $\Pi = (Gen, Enc, Dec)$ *is a IND-CPA secure encryption scheme. Then we can construct an encryption scheme* $\Pi' = (Gen', Enc, Dec')$ *from* Π *such that* Π' *is still IND-CPA secure, and, moreover, for every pair of keys* (pk, sk)*, one can check in polynomial-time whether* $(pk, sk) \in supp(Gen'(1^\eta))$*.*

Proof. Without loss of generality, we may assume that the Gen function on input 1^η uses exactly $p(\eta)$ random bits for some fixed polynomial function p. So we denote by $Gen(1^\eta, r)$ the result of running Gen on input 1^η and using r for randomness ($|r| = p(\eta)$). Now define $Gen'(1^\eta)$ as follow:

$$Gen'(1^\eta, r) = (pk, sk||r) \;\Leftrightarrow\; Gen(1^\eta, r) = (pk, sk)$$

That is, if sk is a private key output by $Gen'(1^\eta)$, its last $p(\eta)$ bits specify the random bits used in outputting sk. The encryption function of Π' is the same as that of Π. As for decryption, we define: $Dec'_{sk||r}(c) = Dec_{sk}(c)$ for $|r| = p(\eta)$. It is obvious that Π' is still IND-CPA secure. However, given a pair of keys $(pk, sk||r)$ with $|r| = p(\eta)$, one can easily test to see if $(pk, sk||r) \in supp(Gen'(1^\eta))$, or not. This completes the proof. □

Theorem 7. *Our co-inductive formal semantics is not computationally complete in the presence of key-cycles if there exists an IND-CPA secure key-bijective encryption scheme.*

Proof. Suppose Π is a IND-CPA secure key-bijective encryption scheme. According to the above lemma, without loss of generality, we may assume that Π has the property that for a given (pk, sk), we can check in a polynomial-time if $(pk, sk) \in^?$

$supp(Gen(1^\eta))$. Thus for any public-key pk, we denote its unique private-key by pk^{-1}, meaning that $(pk, pk^{-1}) \in supp(Gen(1^\eta))$. We prove that there exists expressions m and n such that $m \not\cong n$ but their computational encodings are indistinguishable. Let $m = \{k^{-1}\}_k$ and $n = \{k_1^{-1}\}_k$. We have $pat(m) = \{k^{-1}\}_k$ and $pat(n) = \{\circ\}_k$, and consequently $m \not\cong n$. Now suppose that $\Pi = (Gen, Enc, Dec)$ is a *key-bijective* IND-CPA secure encryption scheme. If $[m]_\Pi \approx [n]_\Pi$ then we are done. Otherwise let $\Pi' = (Gen, Enc', Dec')$ be another encryption scheme which has the same key generation algorithm as Π and its encryption and decryption functions are defined as follow:

$$Enc'(x, pk) = \begin{cases} Enc(x0, pk) & x \neq \langle pk^{-1}, \text{“privkey”} \rangle \\ Enc(1^{|x|+1}, pk) & \text{otherwise} \end{cases}$$

$$Dec'(c, sk) = \begin{cases} x & (x0 = Dec(c, sk)) \\ \langle sk, \text{“privkey”} \rangle & (Dec(c, sk) = 1^\ell) \text{ where } \ell = |\langle sk, \text{“privkey”} \rangle| + 1. \\ \perp & \text{otherwise} \end{cases}$$

As explained before, Π' is also IND-CPA secure. Note that key-bijectiveness here guarantees that the decryption function is well-defined. In other words, if we did not have this condition and sk_1 and sk_2 were both private keys of pk, then $Enc_{pk}(sk_1) = Enc_{pk}(sk_2)$, and thence the decryption function would not have been well-defined. Now note that we have:

$$\begin{aligned} [n]_{\Pi'} &= [\{k_1^{-1}\}_k]_{\Pi'} \\ &= \langle Enc'([k_1^{-1}], \tau(k)), \text{“ciphertext”} \rangle \\ &= \langle Enc([k_1^{-1}]0, \tau(k)), \text{“ciphertext”} \rangle \end{aligned}$$

$$\begin{aligned} [m]_{\Pi'} &= [\{k^{-1}\}_k]'_\Pi \\ &= \langle Enc'([k^{-1}], \tau(k)), \text{“ciphertext”} \rangle \\ &= \langle Enc(1^{|[k^{-1}]|+1}, \tau(k)), \text{“ciphertext”} \rangle \end{aligned}$$

$$= \langle Enc(1^{|[k_1^{-1}]|+1}, \tau(k)), \text{“ciphertext”} \rangle$$

Now since Π is IND-CPA secure, it holds that $Enc(\ 1^{|[k_1^{-1}]|+1}, \tau(k)\ ) \approx Enc(\ [k_1^{-1}]0, \tau(k)\ )$, which implies that $[m]_{\Pi'} \approx [n]_{\Pi'}$, and this completes the proof. □

Note that the counter-example we provided above indicates that if there exists a key-bijective IND-CPA secure encryption scheme, then there exists an IND-CPA secure encryption schemes which is provably *circular secure* in the presence of encryption-cycles of length one. See [13] for the notion of circular-security and its related concepts.

4.3.3 Absence of key-cycles

We first give the characterization of a weakened form of strong public-key indistinguishability of definition 6, where the adversary's oracle calls of the second type are dropped. We call this new indistinguishability notion *weak public-key indistinguishability*:

Definition 11. *(Weak public-key indistinguishability) Let T be a set of public key symbols. Given a public key encryption scheme Π, we say that $[e_1]_\Pi^{\eta,\tau} \approx_w^{O^T} [e_2]_\Pi^{\eta,\tau}$, if for all PPT adversaries $\mathcal{A}$, and random key assignment functions τ, the function:*

$$\begin{aligned}\epsilon(\eta) = \Pr[D(d \leftarrow [e_1]_\Pi^\eta : \mathcal{A}^{O^T}(\eta, d) = 1] \\ - \Pr[D(d \leftarrow [e_2]_\Pi^\eta : \mathcal{A}^{O^T}(\eta, d) = 1]\end{aligned}$$

is negligible; where:

$$O^T(\sigma, pk) = \begin{cases} Dec_{pk^{-1}}(\sigma) & pk \in \tau(T) \\ \perp & otherwise \end{cases}$$

Our computational soundness result, corollary 2, implies that if Π is CCA-2 secure then:

$$pat_C(e_1, T) = pat_C(e_2, T) \;\Rightarrow\; [e_1]_\Pi \approx_C^{T_1^{-1}, T^{-1}} [e_2]_\Pi \qquad (*)$$

where $T_1 = pubkeys(e_1) \cup pubkeys(e_2)$. We prove that under reasonable hypotheses, the converse of (*) also holds. First, we assume that our encryption schemes satisfy a property called *length-revealing*, which intuitively says that a ciphertext reveals the length of its underlying plaintext. This directly captures the assumptions we have already made about our formal semantics:

Definition 12. *An encryption scheme $\Pi = (Gen, Enc, Dec)$ is said to be length-*

revealing if for all security parameters η *and* $pk \in sup(Gen_1(1^\eta))$*, we have:*

$$|m_1| = |m_2| \;\Leftrightarrow\; |Enc_{pk}(m_1)| = |Enc_{pk}(m_2)|$$

Formal cryptography treats formal encryption as a "black-box" whose inversion is possible only by having the right secret key. Moreover, it assumes that deciphering a ciphertext with a wrong key always leads to a failure whose happening is evident to the formal adversary. These assumptions are primarily intended to make the task of *formal verification* of security protocols more manageable. As a result of such an idealized view, the formal adversary is provided with quite strong distinguishing strength. For example, it is able to distinguish between the two expressions (k, k^{-1}) and (k, k_1^{-1}), or more tangibly, between the two expressions $e_1 = (\{m\}_k, k^{-1})$ and $e_2 = (\{m\}_k, k_1^{-1})$. In e_1, the decryption of $\{m\}_k$ under the provided key, i.e. k, succeeds, while in e_2, the decryption fails (and this failure is evident to the adversary). However, such assumptions are not always supported by computational encryption schemes. In effect, it is quite possible that the decryption of a ciphertext under a non-matching decryption key succeeds, and, thence, the computational adversary might not be able to, say, distinguish between the computational interpretations of e_1 and e_2 on the basis of the "failure of decryption under wrong keys". In [29], in order to resolve this discrepancy, they have, further, required the computational encryption scheme to satisfy the *confusion-freeness* property (originally defined in [2]), which informally states that the decryption of a ciphertext, encrypted with a random public-key, under a randomly chosen private-key fails (i.e. results in $\perp$), except with negligible probability. They have also shown that the property of confusion-freeness is provided by a well-known notion of security called *authenticated-encryption* [9]. In our work, instead of using stronger computational security conditions as [29, 22], we hinder such a possibility mentioned above by letting the computational image of a private-key symbol include the computational image of its underlying public-key. That is, we define $[k^{-1}]_\Pi^{\eta,\tau} = \langle \tau(k^{-1}), \tau(k), \text{"privkeys"} \rangle$. This way, the computational adversary is able to distinguish between $[(\{m\}_k,\ k^{-1})]_\Pi^{\eta,\tau}$ and $[(\{m\}_k,\ k_1^{-1})]_\Pi^{\eta,\tau}$, capturing the intuition of formal encryption.

Now we proceed to prove our computational completeness results:

Claim 1. *(*Computational completeness*) We claim that, in the absence of key-cycles, the converse of (*) also holds with respect to any length-revealing encryption scheme; namely, we prove that if* Π *is a length-revealing encryption scheme and*

$T_1 = pubkeys(e_1) \cup pubkeys(e_2)$, *then*

$$[e_1]_\Pi \approx_w^{T^{-1}} [e_2]_\Pi \;\Rightarrow\; pat_C(e_1, T) \cong pat_C(e_2, T)$$

which implies

$$[e_1]_\Pi \approx_C^{T_1, T^{-1}} [e_2]_\Pi \;\Rightarrow\; pat_C(e_1, T) \cong pat_C(e_2, T).$$

As a special case, letting $T = \emptyset$ *we get :*

$$[e_1]_\Pi \approx [e_2]_\Pi \;\Rightarrow\; e_1 \cong e_2$$

Claim 2. *(*Equivalence of weak and strong indistinguishability*) We claim that in the absence of key-cycles and with respect to any CCA-2 secure encryption scheme, the two indistinguishability notions we have introduced (weak and strong) collapse up to renaming of keys. In other words, for CCA-2 secure encryption scheme* Π*, it holds:*

$$[e_1]_\Pi \approx_w^T [e_2]_\Pi \;\Rightarrow\; \exists\sigma\, [e_1]_\Pi \approx_C^{T, T_1} [e_2\sigma]_\Pi$$

where $T_1 = pubkeys(e_1) \cup pubkeys(e_2)$*. Note that we obviously have*

$$[e_1]_\Pi \approx_C^{T_1, T} [e_2]_\Pi \;\Rightarrow\; [e_1]_\Pi \approx_w^T [e_2]_\Pi$$

Thus these two notions of indistinguishability are equivalent up to renaming of keys.

Note that the result of claim 2 directly follows from the result of of claim 1. In order to see this, suppose $[e_1]_\Pi \approx_w^T [e_2]_\Pi$. Now according to the result of claim 1 we have $pat_C(e_1, T^{-1}) \cong pat_C(e_2, T^{-1})$, and consequently for some key-renaming function σ, we have $pat_C(e_1, T^{-1}) = pat_C(e_2\sigma, T^{-1})$. Now according to our computational soundness result we have:

$$pat_C(e_1, T^{-1}) = pat_C(e_2\sigma, T^{-1}) \;\Rightarrow\; [e_1]_\Pi \approx_C^{T_1, T} [e_2\sigma]_\Pi$$

We establish the proof of claim 1 through a sequence of lemmas and theorems. We first show its validity for the special case $T = \emptyset$, and then we show how we can generalize the idea of the proof for any set T.

Theorem 8. *(*Completeness theorem: $T = \emptyset$*) Suppose e_1 and e_2 are two key-acyclic formal expressions, and Π is a length-revealing encryption scheme. If $e_1 \not\cong e_2$, then there exists a PPT adversary $\mathcal{A}$ and a computational pairing function $\langle\rangle$ [2] when used in [.] definition, the two probability distributions $[e_1]^{\eta}_{\Pi}$ and $[e_2]^{\eta}_{\Pi}$ are distinguishable to $\mathcal{A}$ with non-negligible probability.*

We now turn our attention to proving the above theorem. As we pointed out in the introduction of this chapter, the following is a necessary, and still weak, condition for the validity of completeness:

$$struct(e_1) \neq struct(e_2) \Rightarrow |[e_1]| \neq |[e_2]|$$

Thus as the first step, we have to develop a computational pairing function with respect to which the above condition holds. In the following, we show that there exists a pairing function with the required property.

Proposition 2. *There exists a polynomial time computable one-to-one function $\langle ., . \rangle : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ which satisfies the following properties:*

1. $|\langle x, y\rangle| = |\langle z, h\rangle| \Leftrightarrow |x| = |z|$ *and* $|y| = |h|$
2. *The function $h(l_1, l_2) = |\langle x_1, x_2\rangle|$, where $|x_1| = l_1$ and $|x_2| = l_2$ is well-defined, and is the length function of $\langle\rangle$. Moreover, both h and h^{-1} are polynomial-time computable.*

Proof. Define : $f(x, y) = 1^{|x|}0xy$. Obviously f is a one-to-one function, and both f and f^{-1} are polynomial time computable. Moreover, we have: $|f(x, y)| \leq 2|x| + 2|y|$ and $poly(x + y) > f(x, y) > x + y$. Now we define $\langle ., . \rangle$ as follows:

$$\langle x, y\rangle = f(x, y)01^{f(2|x|,2|y|)-|f(x,y)|-1}.$$

Note that $|\langle x, y\rangle| = f(2|x|, 2|y|)$. Since f is a one-to-one function, this implies that:

$$|\langle x, y\rangle| = |\langle z, h\rangle| \Leftrightarrow |x| = |z| \text{ and } |y| = |h|$$

For the second part we have: $h(l_1, l_2) = f(2l_1, 2l_2)$, and thus both h and h^{-1} are polynomial-time computable. This completes the proof. □

[2] It will become clear in the proof of this theorem why we have to put this condition

Henceforth, we assume that the pairing function is the function $\langle ., . \rangle$ introduced in the above proposition. Based on what we have proved so far, we can now give a precise definition of a computational semantics which satisfies condition 4.3.1.

(Computational encoding - Revisited) Recall that every bit-string is tagged with its type (i.e. "Block", "pubkey", ...). We assume that $|\text{"Block"}| = 1, |\text{"pubkey"}| = 2, |\text{"privkey"}| = 3, |\text{"pair"}| = 4$, and $|\text{"ciphertext"}| = 5$. Also we take the function $\langle ., . \rangle$ introduced in the above proposition as our computational pairing function. Also, as explained before, we define $[k^{-1}]^{\eta,\tau}_{\Pi} = \langle \tau(k^{-1}), \tau(k), \text{"privkey"} \rangle$.

Now we can prove the following theorem:

Theorem 9. *Let e_1 and e_2 be two formal expressions. It holds that:*

$$struct(e_1) = struct(e_2) \Leftrightarrow |[e_1]^{\eta}_{\Pi}| = |[e_2]^{\eta}_{\Pi}|.$$

Proof. The proof of the only-if direction ($\Rightarrow$) is straightforward. For the if direction ($\Leftarrow$) note that if two formal expressions are of different types (i.e. have different tags), the length of their computational images is different (this is because of the fact that the function $\langle ., . \rangle$ satisfies the condition specified in proposition 2). Using this fact, we can easily prove the if direction by an structural induction on e_1. We omit the details. □

We now proceed to prove our main theorem of completeness. We take the same approach taken in [29]; we prove that if two formal expressions are not equivalent, then there exists an adversary which can distinguish between their corresponding probability distributions. We establish this by showing that there exists an adversary which can compute $pat_C(e)$ with non-negligible probability from $E \leftarrow [e]^{\eta}_{\Pi}$.

Remark 3. *For the key assignment function τ we have:*

$$pr[\tau(k_1) = \tau(k_2)] = negl(\eta), \text{where } k_1 \neq k_2$$
$$pr[\tau(k_1^{-1}) = \tau(k_2^{-1})] = negl(\eta), \text{where } k_1^{-1} \neq k_2^{-1}$$

We implicitly use these simple facts in our proofs.

Definition 13. *Let $E \in supp[e]^{\eta,\tau}_{\Pi}$ for some pattern e and*

$$T_c = \{[k^{-1}]^{\eta,\tau}_{\Pi} : k^{-1} \in K_{priv}\}$$

We define $\mathcal{F}'_E : \mathcal{P}(T_c) \to \mathcal{P}(T_c)$, the computational key recovery function, *recursively as follow:* ($T'_c \subseteq T_c$)

- *If* $E = \langle pk, \text{“pubkey”} \rangle$, $\mathcal{F}'_E(T'_c) = \emptyset$.
- *If* $E = \langle sk, pk, \text{“privkey”} \rangle$,

$$\mathcal{F}'_E(T'_c) = \langle sk, pk, \text{“privkey”} \rangle$$

- *If* $E = \langle A, B, \text{“pair”} \rangle$, $\mathcal{F}'_E(T'_c) = \mathcal{F}'_A(T'_c) \cup \mathcal{F}'_B(T'_c)$.
- *If* $E = \langle C, pk, \text{“ciphertext”} \rangle$, *there are two cases:*
  1. *If there exists exactly one* sk *for which* $\langle pk, sk, \text{“privkey”} \rangle \in T$, *then*

$$\mathcal{F}'_E(T'_c) = \mathcal{F}'_{Dec(C,sk)}(T'_c)$$

  2. *otherwise:* $\mathcal{F}'_E(T_c) = \emptyset$

Recall the key recovery function $\mathcal{F}$ introduced in the formal setting. The following lemma shows that $\mathcal{F}'$, the computational key recovery function, is in fact a computational counterpart of F. In other words, if $T \subseteq K_{priv}$, $T_c = [T]^{\eta,\tau}_{\Pi}$, and $E \leftarrow [e]^{\eta,\tau}_{\Pi}$, then $[\mathcal{F}_e(T)]^{\eta,\tau}_{\Pi} = \mathcal{F}'_E(T_c)$ with overwhelming probability.

Lemma 9. *For* $e \in Pat$, *let* $E \leftarrow [e]^{\eta,\tau}_{\Pi}$, $T_c = [T]^{\eta,\tau}_{\Pi}$, *and* $T'_c = [\mathcal{F}_e(T)]^{\eta,\tau}_{\Pi}$. *We have :*

$$pr[T'_c \neq \mathcal{F}'_E(T_c)] \leq negl(\eta).$$

Proof. We prove this by structural induction on e. For the base case, if e is a block symbol or a key symbol(public or private), it obviously holds that: $pr[T'_c \neq \mathcal{F}'_E(T_c)] = 0$. Now we have:

- if $e = (e_1, e_2)$ and $E = (E_1, E_2)$, we have: $pr[\, \mathcal{F}'_E(T_c) \neq [\mathcal{F}_e(T)] \,] = pr[\, \mathcal{F}'_{E_1}(T_c) \neq [\mathcal{F}_{e_1}(T)]$ *or* $\mathcal{F}'_{E_2}(T_c) \neq [\mathcal{F}_{e_2}(T)] \,] \leq pr[\, \mathcal{F}'_{E_1}(T_c) \neq [\mathcal{F}_{e_1}(T)\,] + pr[\, \mathcal{F}'_{E_2}(T_c) \neq [\mathcal{F}_{e_2}(T)] \,] \leq negl(\eta) + negl(\eta) = negl(\eta)$.

- if $e = \{e_1\}_k$ and $E = \langle c, pk, \text{“ciphertext”} \rangle$, then there are two cases(assume that E_1 is the computational image of e_1 in E):

1. $k^{-1} \notin T$. Therefore, we have: $\mathcal{F}_e(T) = \emptyset$, and consequently:

$$pr[\, \mathcal{F}'_E(T_c) \neq [\mathcal{F}_e(T)]\,] = pr[\, \mathcal{F}'_E(T_c) \neq \emptyset]$$
$$= pr[\, \exists k_1^{-1} \in T \; such \; that \; \tau(k_1) = \tau(k)\,] \leq negl(\eta)\ .$$

2. $k^{-1} \in T$. It holds that $\mathcal{F}_e(T) = \mathcal{F}_{e_1}(T)$, and consequently :

$$pr[\, \mathcal{F}'_E(T_c) \neq [\mathcal{F}_e(T)]\,] = pr[\, \mathcal{F}'_E(T_c) \neq [\mathcal{F}_{e_1}(T)]\,]$$
$$\leq pr[\, \exists k_1^{-1} \in T \; such \; that \; k_1^{-1} \neq k^{-1} \;\&\; \tau(k_1) = \tau(k)\,] +$$
$$pr[\, \mathcal{F}'_E(T_c) \neq [\mathcal{F}_{e_1}(T)] \mid \nexists k_1^{-1} \in T \; such \; that \; k_1^{-1} \neq k^{-1} \;\&\; \tau(k_1) = \tau(k)\,]$$
$$\leq negl(\eta) + pr[\, \mathcal{F}'_{E_1}(T_c) \neq [\mathcal{F}_{e_1}(T)]\,] \leq negl(\eta) + negl(\eta) = negl(\eta).$$

□

Note that since we are dealing with key-acyclic expressions, according to lemma 2, it holds that $FIX(\mathcal{F}_e) = fix(\mathcal{F}_e)$. Therefore, from now on, we focus our attention on the least fixed point. In particular, having defined the computational key recovery function, we can define its least fixed point in the usual way (i.e. the least fixed point of $\mathcal{F}'_E$ is the smallest set T_c for which $\mathcal{F}'_E(T_c) = T_c$.) Suppose $e \in Exp$ and $E \leftarrow [e]^{\eta}_{\Pi}$ and $n = |privkeys(e)|$. Likewise lemma 1, it can be easily verified that the least fixed point of $\mathcal{F}'_E$ admits the following formula:

$$fix(\mathcal{F}'_E) = \bigcup_i \mathcal{F}'^i_E(\emptyset) = \mathcal{F}'^n_E(\emptyset).$$

Using the above formula, it can be seen that $fix(\mathcal{F}'_E)$ is polynomial-time computable. The following lemma relates the computational least fixed point to the formal least fixed point:

Lemma 10. *Let* $E \leftarrow [e]^{\eta,\tau}_{\Pi}$ *, and* $T = [fix(\mathcal{F}_e)]^{\eta,\tau}_{\Pi}$*. It holds that:*

$$pr[fix(\mathcal{F}'_E) \neq T] \leq negl(\eta).$$

Proof. Let $n = |privkeys(e)|$. Note that $fix(\mathcal{F}'_E) = \mathcal{F}'^n_E(\emptyset)$ and $fix(\mathcal{F}_e) = F^n_e(\emptyset)$.

We have

$$\begin{aligned} pr[fix(\mathcal{F}'_E) \neq T] &\leq pr[\exists i \leq n : [\mathcal{F}_e^i(\emptyset)]_\Pi^\eta \neq \mathcal{F'}_E^i(\emptyset)] \\ &\leq n \cdot negl(\eta) \\ &= negl(\eta) \end{aligned}$$

□

In theorem 9, we showed that under our proposed computational semantics, there exists a one-to-one correspondence between the length of a computational message and its structure. In the following lemma, we show that this correspondence is computationally constructible. That is, there exists a polynomial time adversary which can recover the structure of a computational message from its computational length.

Lemma 11. *(*Structure recovery from computational length*) Let $e \in Exp$, $\Pi = (Gen, Enc, Dec)$ be a length-revealing encryption scheme, and $E \leftarrow [e]_\Pi^{\eta,\tau}$. There exists a polynomial time function which can compute the structure of e when given $|E|$ and η.*

Proof. Since the adversary is given η, it can run the key generation algorithm to obtain a pair of public/private keys (k, k^{-1}). Let $f : \mathbb{N} \to \mathbb{N}$ be the length function of Enc_k which is by assumption one-to-one. Since the encryption function is publicly known, every adversary is able to computer f^{-1}. Also, let h be the length function of $\langle ., . \rangle$, namely if $|x_1| = l_1$ and $|x_2| = l_2$, then $h(l_1, l_2) = |\langle x_1, x_2 \rangle|$. Recall from proposition 2 that both h and $h^{-1}$are polynomial-time computable. Also, we write $h(l_1, l_2..., l_n)$ for $h(..., h(h(l_1, l_2), l_3), ..., l_n)$. We now present an algorithm which recovers the structure of a message from its length:

algorithm $STRUCT\_REC(|E|)$
  if $|E| = h(r, |\text{“Block”}|)$
   return $\square$
  if $|E| = h(|k|, |\text{“pubkey”}|)$
   return $\circ$
  if $|E| = h(|k^{-1}|, |k|, |\text{“privkey”}|)$
   return $\circ_p$
  if $|E| = h(x_1, x_2, |\text{“pair”}|)$

return $(STRUCT\_REC(x_1), STRUCT\_REC(x_2))$
if $|E| = h(x, |k|, |\text{“ciphertext”}|)$
return $\{STRUCT\_REC(f^{-1}(x))\}_\circ$

Recall that the length of tags are different. Now using the fact that h is one-to-one, it can be seen by an straightforward induction on the structure of e that it holds that:

$$pr[STRUCT\_REC(|E|) = struct(e)] = 1.$$

We omit the details of the proof. □

We now have the necessary tools to present an algorithm which takes as input a bit-string and outputs the pattern of its underlying formal expression. We prove that the algorithm succeeds with probability $1 - negl(\eta)$.

Lemma 12. *(*pattern recovery*) Let $e \in Exp$, $T \subseteq K_{priv}$, $\Pi = (Gen, Enc, Dec)$ be a length-revealing encryption scheme, and $E \leftarrow [e]^{\eta,\tau}_\Pi$. There exists a PPT adversary $\mathcal{A}$ which can computer $pat(e)$ [3] when given E and η with probability $1 - negl(\eta)$.*

Proof. In the proof of this theorem, we use K, $K_1, \ldots,$ for referring to key values and k, $k_1, \ldots,$ for denoting key symbols. As explained in remark 3, we can safely assume that every two public (private) key symbols receive different key values from the key assignment function τ. Also, let $f : \mathbb{N} \to \mathbb{N}$ be the length function of Enc_K for $K \in supp(Gen_1(1^\eta))$, and φ be a function which is to assign a public (resp. private) key symbol to every public (resp. private) key value. Also, assume that Pub and $Priv$ are two lists of public/private key values which are initially empty. The adversary $\mathcal{A}$ first computes $fix(\mathcal{F}'_E)$, and then calls $INITIALIZATION(fix_c(\mathcal{F}'_E))$, defined as follows:

algorithm $INITIALIZATION(R)$
for every $\langle K^{-1}, K, \text{“privkey”}\rangle \in R$ **do**
add K to Pub and K^{-1} to $Priv$,
create two new key symbols k and k^{-1},
assign: $\varphi(K) = k$ and $\varphi(K^{-1}) = k^{-1}$
$PAT\_REC(E)$

[3] since by assumption e is key-acyclic, we have $pat_I(e) = pat_C(e)$, and for simplicity, we refer to the pattern of e by $pat(e)$.

algorithm $PAT\_REC(E)$
  if $E = \langle E_1, \text{“Block”} \rangle$
    return E_1
  if $E = \langle K^{-1}, K, \text{‘privkey”} \rangle$
    return $\varphi(K^{-1})$
  if $E = \langle K, \text{“pubkey”} \rangle$
    if $K \notin Pub$
      add K to Pub, and create a new key symbol k,
      and assign $\varphi(K) = k$,
    return $\varphi(K)$
  if $E = \langle E_1, E_2, \text{“pair”} \rangle$
    return $(PAT\_REC(E_1), PAT\_REC(E_2))$
  if $E = \langle C, K, \text{″}ciphertext\text{″} \rangle$
    if there exists a private key value K^{-1} for which
      $\langle K^{-1}, K, \text{“privkey”} \rangle \in fix_c(\mathcal{F}'_E)$
      return $\{PAT\_REC(Dec_{K^{-1}}(E))\}_{\varphi(K)}$
    otherwise
      if $K \notin Pub$
        add K to Pub, and create a new key symbol k,
        and assign $\varphi(K) = k$,
      return $\{STRUCT\_REC(f^{-1}(|E|))\}_{\varphi(K)}$

Now if we define the key bijection function $\sigma(k) = \varphi(\tau(k))$, it can be easily seen by an structural induction on e that $pr[pat(e) \neq PAT\_REC(E)\sigma] = negl(\eta)$. We omit the details of the proof. □

Now we can prove our two main theorems we presented earlier.

(Proof of theorem 8) The proof of this theorem follows immediately from the pattern recovery lemma. Suppose $E_1 \leftarrow [e_1]^{\eta,\tau}_{\Pi}$ and $E_2 \leftarrow [e_2]^{\eta,\tau}_{\Pi}$. Note that since $e_1 \ncong e_2$, it follows that $pat(e_1) \ncong pat(e_2)$, and consequently, $pr[PAT\_REC(E_1) \cong pat(e_2)] = negl(\eta)$. On the other hand, we have : $pr[PAT\_REC(E_2) \cong pat(e_2)] = 1 - negl(\eta)$. This observation immediately leads to a distinguisher $\mathcal{A}$ which on input E_i computes $pat(e_i)$ by calling $PAT\_REC(E_i)$, and compares it against $pat(e_2)$. If they match up to key renaming, it returns one, otherwise it returns zero. Obviously $\mathcal{A}$ works in polynomial time and is able to distinguish between the two probability distributions with non-negligible probability. ■

(Proof of claim 1) (Sketch) We just need to prove that

$$pat(e_1, T) \ncong pat(e_2, T) \Rightarrow [e_1]_\Pi \not\approx_w^{T^{-1}} [e_2].$$

Suppose $pat(e_1, T) \ncong pat(e_2, T)$. According to theorem 8, we have $[pat(e_1, T)]_\Pi \not\approx_w [pat(e_2, T)]_\Pi$. Thus it remains to prove that

$$[pat(e_1, T)]_\Pi \not\approx_w [pat(e_2, T)]_\Pi \Rightarrow [e_1]_\Pi \not\approx_w^{T^{-1}} [e_2].$$

This is easy to see based on what we have proved before. Suppose there exists an adversary $\mathcal{A}$ which can distinguish $[pat(e_1, T)]_\Pi$ from $[pat(e_2, T)]_\Pi$. From $\mathcal{A}$, we can now construct another adversary $\mathcal{B}^{O^{T^{-1}}}$ which can distinguish $[e_1]_\Pi$ from $[e_2]_\Pi$. The idea of construction is simple and we just outline the main idea and leave out the details. Suppose $\mathcal{B}^{O^{T^{-1}}}$ is given E_i which is a sample from $[e_i]_\Pi$. Using its oracle access, it can turn E_i into a sample D_i from $[pat(e_i, T)]_\Pi$, and give D_i to $\mathcal{A}$, and output whatever $\mathcal{A}$ outputs. In more detail, consider that $[pat(e_i, T)]_\Pi = [p(e_i, fix(\mathcal{F}_{e_i T}))]_\Pi$. Now letting $T_1 = fix(\mathcal{F}_{e_i T})$, note that $\mathcal{B}$, using its oracle access, can obtain the key values of all key symbols $T_1 - T$. For any $k \in T$, although it might now have its key value, it can ask its oracle to decrypt with respect to it. Now $\mathcal{B}$ can construct D_i as follow: for every $\langle c, pk, \text{“ciphertext”} \rangle$ which is derivable from E_i and is no more decryptable, replace c with $Enc_{pk}(0^d)$, where d is the length of the underlying plaintext of c. ■

Chapter 5

Conclusion and Future Work

We have considered the relation between Dolev-Yao and computational models of cryptography in the co-inductive setting, and in the context of non-malleability and indistinguishability. In addition to formulating the indistinguishability notion of [21, 20] in our co-inductive framework, we gave a co-inductive definition to Herzog's non-malleability notion, and we proved that both properties are satisfied in by IND-CCA2 encryption schemes. Note that Corollary 6 immediately implies that, in both the inductive and co-inductive settings, any security notion weaker than IND-CCA1 does not provide Dolev-Yao non-malleability. One extension of this work would to investigate the relationship between Dolev-Yao non-malleability and those notions not implied by IND-CCA1 secrecy (good candidates are NM-CCA1 and NM-CPA. See for a discussion on the relative strength of security notions in public-key cryptography). Moreover, as mentioned before, our non-malleability notion may be strengthened by giving the computational adversary a decryption oracle as follows:

Definition 14. *(Strong Dolev-Yao non-malleability) The encryption scheme* $\Pi = (Gen, Enc, Dec)$ *provides* co-inductive (resp. inductive) strong Dolev-Yao public-key non-malleability *if for all* $S \subseteq Exp$, $e \notin Closure_C(S)$ *(resp.* $e \notin Closure_I(S)$ *), and PPT adversaries* $\mathcal{A}$*, the following function:*

$$\epsilon(\eta) = \Pr[\, d_1 \leftarrow [S]^{\eta,\tau}_{\Pi};\ d_2 \leftarrow \mathcal{A}^{pbK^{\tau}_{\eta}(.),\mathcal{O}_{d_1}}(1^{\eta}, d_1) :\ d_2 \in supp[e]^{\eta,\tau}_{\Pi}\,]$$

is negligible; where $pbK^{\tau}_{\eta}(k)$ *returns* $\tau(k)$*, and* $\mathcal{O}_{d_1}(\sigma, pk)$ *returns* $Dec_{pk^{-1}}(\sigma)$ *if* $pk = \tau(k)$ *for* $k \in pubkeys(S)$ *and* $\langle \sigma, pk, \text{"cipher"} \rangle \notin Vis_{d_1}(\tau(FIX^{-1}(\mathcal{F}_S)))$*, and returns* $\perp$ *otherwise.*

It would be interesting to examine the relationship between the strong indistinguishability notion (definition 7) and this stronger notion of non-malleability. In particular, our counter-example presented in theorem 5 does not work for this stronger notion of non-malleability.

We remark that the definition of Dolev-Yao non-malleability (definition 10) can be generalized, in another way, to yield a stronger notion in which the adversary's goal is to produce the computational interpretation of any message outside the closure set, not necessarily a fixed given message [20]. The reason that we chose this current notion of non-malleability (which was also used in [21]), and not the stronger version, is because of the simplicity of presentation. Nevertheless, all the results we have established for non-malleability easily extend for this stronger definition (proofs of theorems 6 and 5 do also work for this stronger definition. Also the same proof technique as [20] can be used to show that co-inductive public-key indistinguishability implies strong Dolev-Yao non-malleability, and consequently Theorem 4 follows for this stronger definition).

Lastly, it would be interesting to investigate how the co-inductive approach can be used to prove the computational soundness results in the case of active adversaries (e.g, [30, 23]).

Bibliography

[1] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proceedings of the 4th ACM conference on Computer and communications security*, CCS '97, pages 36–47, New York, NY, USA, 1997. ACM.

[2] Martín Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation. In *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software*, TACS '01, pages 82–94, London, UK, 2001. Springer-Verlag.

[3] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, TCS '00, pages 3–22, London, UK, 2000. Springer-Verlag.

[4] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage. *J. Comput. Secur.*, 17:737–797, October 2009.

[5] Pedro Adao, Gergei Bana, and Andre Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 170–184, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Pedro Ado, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness of formal encryption in the presence of key-cycles. In *In Proc. 10th European Symposium on Research in Computer Security (ESORICS05), volume 3679 of LNCS*, pages 374–396. Springer, 2005.

[7] Gergely Bana. *Soundness and completeness of formal logics of symmetric encryption*. PhD thesis, Philadelphia, PA, USA, 2004. AAI3137979.

[8] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: security proofs and improvements. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 259–274, Berlin, Heidelberg, 2000. Springer-Verlag.

[9] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21:469–491, September 2008.

[10] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, SAC '02, pages 62–75, London, UK, 2003. Springer-Verlag.

[11] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman, 2008.

[12] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8:18–36, February 1990.

[13] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 93–118, London, UK, 2001. Springer-Verlag.

[14] D. Dolev and A. C. Yao. On the security of public key protocols. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:350–357, 1981.

[15] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 542–552, New York, NY, USA, 1991. ACM.

[16] Flavio D. Garcia and Peter van Rossum. Sound and complete computational interpretation of symbolic hashes in the standard model. *Theor. Comput. Sci.*, 394:112–133, March 2008.

[17] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[18] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[19] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[20] Jonathan Herzog. *Computational Soundness for Standard Assumptions of Formal Cryptography*. PhD thesis, Massachusetts Institute of Technology, May 2004.

[21] Jonathan Herzog. A computational interpretation of dolev-yao adversaries. *Theor. Comput. Sci.*, 340:57–81, June 2005.

[22] Omer Horvitz and Virgil Gligor. Weak key authenticity and the computational completeness of formal encryption. In *In Crypto 2003, volume 2729 of LNCS*, pages 530–547. Springer-Verlag, 2003.

[23] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *ESOP*, pages 172–185, 2005.

[24] Peeter Laud. Encryption cycles and two views of cryptography. In *In NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems (Karlstad University Studies 2002:31*, pages 85–100, 2002.

[25] Peeter Laud and Ricardo Corin. Sound computational interpretation of formal encryption with composed keys. In *In Information Security and Cryptology - ICISC 2003, 6th International Conference, LNCS*, pages 55–66. Springer-Verlag, 2003.

[26] Daniele Micciancio. Pseudo-randomness and partial information in symbolic security analysis, 2009.

[27] Daniele Micciancio. Computational soundness, co-induction, and encryption cycles. In *EUROCRYPT*, volume 6110, page 367, 2010.

[28] Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In *In Proc. 2nd Theory of Cryptography Conference (TCC05), volume 3378 of LNCS*, pages 169–187. Springer, 2005.

[29] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. *J. Comput. Secur.*, 12:99–129, January 2004.

[30] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *In Proc. 1st Theory of Cryptography Conference (TCC), volume 2951 of LNCS*, pages 133–151. Springer, 2004.

[31] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 433–444, London, UK, 1992. Springer-Verlag.