SOUNDNESS AND COMPLETENESS OF FORMAL LOGICS OF SYMMETRIC ENCRYPTION

Gergely Bana

A DISSERTATION

in

Mathematics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2005

Andre Scedrov, Supervisor of Dissertation

David Harbater, Graduate Group Chairperson

Acknowledgements

I would like to thank Andre Scedrov for his tremendous support as a thesis advisor, and also for introducing me to the subject and to the information security and cryptography community. I would also like to thank Pedro Adão from the University of Lisbon for the great hours of research we spent on the subject, and for reading the manuscript and providing valuable suggestions. Finally, many thanks are due to Richard V. Kadison for the lunches, and for all that I learned from him in math and in life matters as well.

During the work on this thesis I was supported by ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795, with additional support by NSF Grant CCR-0098096.

ABSTRACT

SOUNDNESS AND COMPLETENESS OF FORMAL LOGICS OF SYMMETRIC ENCRYPTION

Gergely Bana

Supervisor: Andre Scedrov

In the last two decades, two major directions in cryptography have developed: formal and computational. The formal approach uses simple, manageable formal languages to describe cryptographic protocols; this approach is amenable to automatization, suitable for computer tools, but its accuracy is often unclear. The computational approach is harder to handle mathematically, involves probability theory and considers limits in computing power; proofs are done by hand, but it is more accurate, hence widely accepted.

Much effort has been made to bridge the gap between the two approaches, including the work of Martin Abadi and Philip Rogaway who considered a formal logic of symmetric encryption and its interpretations in cryptosystems based on computational complexity. The Abadi-Rogaway setting has three important ingredients: a formal language along with an equivalence notion of formal expressions, a computational cryptosystem with the notion of computational equivalence of ensembles of random distributions, and an interpreting function that assigns to each formal expression an ensemble of distributions. We say that the interpretation satisfies soundness if equivalence of formal expressions implies computational equivalence of their interpretations, and satisfies completeness if computational equivalence of the interpretations requires equivalence of the expressions.

We consider expansions of the Abadi-Rogaway logic of indistinguishability of formal cryptographic expressions. The formal language of this logic uses a box as notation for indecipherable strings, through which formal equivalence is defined. We expand the logic by considering different kinds of boxes corresponding to equivalence classes of formal ciphers. We consider not only computational, but also purely probabilistic, information-theoretic interpretations. We present a general, systematic treatment of the expansions of the logic for symmetric encryption. We establish general soundness and completeness theorems for the interpretations. We also present applications to specific settings not covered in earlier works: a purely probabilistic one that in-

terprets formal expressions in One-Time Pad, and computational settings of the so-called type 2 (which-key revealing) cryptosystems based on computational complexity.

Contents

Introduction

Designing and verifying security protocols are complex problems; certain level of idealization is needed in order to provide manageable mathematical treatment of the protocols and the notion of security. Idealizations necessarily omit some properties of the real system, which might lead to leaks in the security. Even if the protocols themselves are quite simple – which is often the case –, the security properties that they are supposed to achieve might be rather subtle, hard to formulate, and checking whether they really achieve them may be an almost impossible task. Difficulties typically arise from subtleties of the cryptographic primitives themselves or while combining them; security protocols are required to work properly when multiple instances are carried out in parallel, in which case a malicious intruder may combine data from separate sessions in order to confuse honest participants.

A number of methods, and different levels of idealizations are used for analyzing security protocols, the two main being a highly abstract treatment with the help of formal logic, and a more detailed description using complexity and probability theory. In the former, cryptographic operations are modeled as functions on a space of symbolic (formal) expressions and their security properties are also treated formally. Examples are [1, 9, 15, 28, 27, 31, 43, 18, 41, 13, 16]. In the latter, cryptographic operations act on strings of bits, and their security properties are defined in terms of probability and computational complexity. Examples for this treatment are [7, 20, 22, 21, 53, 8]. The first approach has been labeled in the literature as *formal view*, whereas the second as *computational view*.

The computational view gives a more detailed description of cryptographic operations, taking limited computing power into account; probability plays an important role as well. "Good protocols are those in which adversaries cannot do something bad too often and efficiently enough"

[3]. Keys, plaintexts, ciphers are all strings of bits, encryption, decryption and adversaries are all probabilistic algorithms, and a mathematically well-defined notion of *computability in polynomial time* is imposed on all these algorithms. This view originates from the work of Blum, Goldwasser, Micali and Yao in [8, 53, 21]. A major achievement of this approach has been that common notions of security such as secrecy, authentication, etc. were given mathematically precise definition, hence clarifying them and making them amenable to mathematical analysis. On the other hand, the detailed and less structured nature of this view makes analyzing more complex protocols a very hard task, which calls for a higher level, more abstract treatment.

Formal methods were first introduced in the works of Dolev, Yao, DeMillo, Lynch, Merritt, Millen, Clark, Freedman, Kemmerer, Burrows, Abadi, and Needham, and Meadows in [13, 15, 41, 9, 27, 37], with many different approaches and techniques. The main formal approaches include specialized logics such as BAN logic, [9, 17], special-purpose tools designed for security protocol analysis, [28], theorem-proving [46, 45] and model-checking methods using several general purpose tools described in [31, 38, 43, 49, 50]; multiset rewriting framework and strand spaces framework [12, 18, 16]. Although these methods differ significantly some ways, many of them rely on the same basic assumptions about how an adversary may interact with the protocol. In the common model, largely derived from [14] and suggestions found in [44], a protocol adversary is allowed to choose among possible actions nondeterministically, but the set of messages he may use to interfere with a protocol must be restricted severely. So, although the idealized assumptions make protocol analysis tractable, they also make it possible to "verify" protocols that are in fact susceptible to simple attacks that lie outside the adversary model. It is therefore necessary to establish criteria about the limitations and applicability of formal methods. It is hoped that clarifying the relationship between the formal and the computational views will lead to a better understanding of this problem.

There have been several research efforts recently to relate the highly idealized formal model of cryptographic techniques and the computational one based on probabilistic polynomial-time computability, including [3, 2, 40, 29, 4, 6, 5, 11, 10, 30, 42, 32, 47, 48, 24]. These efforts are developing rigorous mathematical treatment of the relationship between the two models. It is hoped that they will eventually lead to a new generation of "high fidelity" automated tools for security analysis, which will be able to express and implement the methods and concepts of

modern cryptography.

Before we go into the details of the relationship of the two model, we discuss a third model as well. This is called information-theoretic view. While computational security of a cryptographic system relies on the computational infeasibility of breaking it, information-theoretic security requires the system to be theoretically unbreakable. This latter condition of course is stronger, and it does not rely on the assumption of computational security (namely that there are problems which are "hard" to solve which has not been proved yet). The origins of this approach go back to the very foundations of cryptography as a science, namely, to Shannon's 1949 paper on the communication theory of secrecy systems [52], which was a companion paper to [51], where he laid the foundations of information theory. However, Shannon's proof that perfect secrecy requires a secret key of at least the length of the plaintext is often considered as an evidence that perfect security can never be practical. But, recent research shows (see e.g. [35, 33, 19, 36, 34]), that there are possibilities for usful applications of information-theory in cryptography. Therefore, in this work, we also explore information-theoretic interpretations of the formal cryptographic expressions.

Previous Work

The original approach of Abadi and Rogaway in [3], of which in this thesis we attempt to give a fairly complete analysis, uses a simple formal structure by building messages from formal keys and bits via repeated pairing and encrypting, constructing a set of formal expressions this way. These formal expressions are then interpreted in a computational framework of symmetric encryptions. Via this interpretation, an ensemble of probability distributions (parametrized by a security parameter) on the set of finite bit strings is assigned to each formal expression. In each of the formal and the computational view, a notion of equivalence expresses security. In the formal view, equivalence of formal expressions are defined; in the computational, equivalence of ensembles of probability distributions. The question is, what happens to the equivalence through the interpretation. If it is true that equivalence of any two formal expressions imply computational equivalence of their interpretations, then we say soundness holds. If the other direction is true, namely, when computational equivalence of the interpretations of any two formal expressions implies that the formal expressions are equivalent too, we then say that completeness holds.

Soundness of their formal logic and its interpretation was proved by Abadi and Rogaway in [3] for the case when the computational encryption scheme in which the formal expressions are interpreted is so-called type-0, which essentially means that no partial information is revealed about the encrypted plaintext and the encrypting keys (hence it is impossible to detect when the encrypting keys or the plaintext is repeated). A little later, in [39], Micciancio and Warinschi showed completeness for this same situation, but requiring also something they called "confusion freeness" of an encryption scheme, which requirement ensures that decrypting with the wrong key be detectable. In this same paper, the authors also indicated how to expand the Abadi-Rogaway formalism when length of the plaintext might be revealed. The condition of confusion freeness was relaxed along with a new completeness proof method by Horvitz and Gligor in [26] replacing confusion freeness with "weak confusion freeness". Extensions of the method includes [40] of Micciancio and Warinschi, where they considered public-key encryption, Laud and Corin's [29] for composite keys, the works of Herzog, Liskov and Micali [25, 24] for plaintext-aware cryptosystems, etc. The connection between symbolic and information-theoretic view was also explored in [23] by Guttman, Thayer and Zuck.

Our Work

We consider the original approach of Abadi and Rogaway, and provide a more complete, more general, and systematic treatment of it, hoping that it will serve as inspiration for dealing with more complex formal cryptographic languages and their interpretations in similar generality.

Our work extends applicability of the AR language in two directions, which are then organized under a single formalism. On one hand, besides interpretations in computational frameworks, we also consider interpretations for purely probabilistic, *information-theoretic* encryption schemes. On the other hand, by expanding the AR language without changing its basic framework, we show how to adjust the formal notion of equivalence in order to maintain soundness and completeness when the encryption scheme that hosts the interpretation (computational of information-theoretic) is such that it leaks partial information.

In order to extend the applicability of the AR language, we change the notion of their formal equivalence The language of the Abadi-Rogaway logic uses a box as formal notation for undecryptable expressions, and equivalence of formal expressions (which expresses the notion of

security) is defined with the help of this box. In the expanded formalism that we adopt, different kinds of boxes are allowed, which - loosely speaking - correspond to different kinds of undecryptable expressions. (This idea is not new, Micciancio and Warinschi already used it in [39] to give a treatment of encryption schemes that reveal the length of the plaintext.) We establish soundness and completeness for two specific interpretations. We also give an abstract treatment about how to handle the different boxes in general, and general soundness and completeness theorems are also established.

The specific interpretations that are discussed in detail are the following: an information-theoretic one that interprets formal expressions in One-Time Pad, and another one in the so-called type-2 (which-key revealing) cryptosystems based on computational complexity. These two examples shed light on the expansions of the AR logic and on he interpretations from two different angles, and they prepare us for the general treatment.

The name type-2 cryptosystem stands for an encryption scheme that might reveal when the same key is used to encrypt twice. We first show soundness of the Abadi-Rogaway interpretation for an expanded logic that includes a separate box for each key in this case. We also show completeness of this such expanded logic for weakly confusion-free, strictly which-key revealing cryptosystems. By "strictly which-key revealing", we mean that there exists a probabilistic polynomial-time adversary that can distinguish two pairs of oracles, one always encrypting with the same key and the other encrypting with different keys. For completeness, we also assume that decryption with the wrong key is computationally distinguishable from the correct decryption; this is what Horvitz and Gligor called "weak confusion-freeness" in [26].

In the case of One-Time Pad (One-Time Pad means that for a plaintext of a given length, a key of the same length is generated uniformly, and XOR'ed with the plaintext), a natural expansion of the Abadi-Rogaway logic involves boxes indexed by the length of the encrypted message (and the matching length of the encrypting key; a formal length-function is introduced). On one hand, we define the formal equivalence of expressions with the help of such boxes. On the other hand, we postulate One-Time-Pad realizations of two formal expressions to be equivalent, if and only if their probability distributions are identical. We show both soundness and completeness for this interpretation, with a specific implementation of the OTP.

In the general treatment, we index the boxes with equivalence classes of formal ciphers, where

the equivalence relation expresses the inability of an adversary to distinguish between ciphers. This equivalence relation must be chosen keeping in mind the security level of the encryption scheme that will host the interpretation. For example, in case of One-Time Pad, since the cryptosystem reveals length, this equivalence relation will corresponds to "same length" of formal ciphers. When we interpret the formal expressions in type-2 cryptosystems, then the equivalence relation will render ciphers encrypted with the same key equivalent. If the scheme reveals both key and length, the the equivalence relation has to be defined on formal ciphers so that equivalence holds iff two ciphers have the same lengths and are encrypted with the same key, and so on.

A general probabilistic framework is also introduced which contains computational and purely probabilistic treatments as special cases. The advantage of this is that there is no need to formulate general statements twice when they are true for both computational and information-theoretic models. In the literature, when computational complexity is considered, probability distributions are indexed with the security parameter, so ensembles of distributions are dealt with; in information-theoretic models, there is no notion of security parameter, and therefore, no ensembles of distributions, only single distributions are handled. To overcome this discrepancy, we show how to "hide" the security parameter, and so to deal with only single distributions in the computational view as well.

Finally, we prove a general soundness and a general completeness theorem. These theorems essentially claim that if soundness or completeness holds for a certain subset of the formal expressions, then soundness or completeness is valid for all expressions. As it is expectable, it is necessary to assume soundness for a greater subset of subexpressions than for completeness in order to derive the theorems. The reason is that the probabilistic view is a much more detailed description: Indistinguishability of distributions of two n-tuples of random variables does not follow from the fact that each two corresponding *pairs* in the n-tuples are indistinguishable (not even if they are all identical); whereas, equivalence of two n-tuples of formal expressions can be built up from pairwise equivalence.

Layout

The dissertation is structured the following way: In section 1.1 we review first the formal language of Abadi and Rogaway, how formal expressions are built, and how their equivalence is defined;

then, we introduce an abstract treatment of the expansions, and prove a few propositions about the notions we introduce there. These propositions are essential for the general soundness and completeness results.

In section 1.2, we review the definition of symmetric encryption schemes and their security in the computational framework, and in subsection 1.2.3, we explain how to hide eta and make the computational formalism compatible with the information-theoretic one. In section 1.3, we present a specific implementation of the One-Time Pad, and in 1.4, we show how to handle computational and information-theoretic encryption schemes together. The first four sections of chapter 2 are devoted to the interpretations and soundness theorems for type-2 encryptions and for the One-Time Pad. In subsection 2.2, besides proving soundness for type-2 systems, we also present a detailed example showing the major steps of the proof. Section 2.5, is devoted to the general interpretation, and the general soundness theorem, together with examples, such as type-3 cryptosystems.

Chapter 3 is devoted to our completeness results. First we need to formalize mathematically the process of decrypting everything that is possible in a sample from the interpretation of a formal expression. This is done in section 3.1, and section 3.2 is the one with the completeness theorems for type-2 schemes, One-Time Pad, and the general case. Again, there is an elaborate concrete example both in section 3.1 and section 3.2 following through all the steps of the theorems there.

Chapter 1

Three Views of Cryptography

1.1 The Abadi-Rogaway Formal Language

1.1.1 The Formal Model for Messages: Expressions

Abadi and Rogaway introduced a simple formal language to model symmetric cryptographic protocols. Although this language is too simple to describe realistic situations, it is very suitable to isolate the subtleties of the relationship between the formal and the computational treatments of cryptography. In this formal setting, a set of *expressions* correspond to the messages that are transmitted during a cryptographic protocol. *Encryption* operates on the set of expressions, resulting new expressions. All expressions are built from keys and blocks of bits via pairing and encryption. Accordingly, the formal definition of expressions is the following.

Definition 1.1 (Expression)**.** Let **Keys** be a discrete set of symbols, namely,

$$\mathbf{Keys} := \{K_1, K_2, K_3, ...\}.$$

Let **Blocks** be a nonempty subset of finite bit-strings:

$$\mathbf{Blocks} \subseteq \{0,1\}^*.$$

We define the *set of expressions*, **Exp**, by the grammar:

$M, N ::=$	*expressions*
K	key (for $K \in \mathbf{Keys}$)
B	block (for $B \in \mathbf{Blocks}$)
(M, N)	pair
$\{M\}_K$	encryption (for $K \in \mathbf{Keys}$)

The set of expressions are denoted by **Exp**. We will denote by $Keys(M)$ the set of all keys occurring in M. Let **Ciphers** denote the set of all encryptions. □

The set **Blocks** is to model a fixed set of strings of messages, whereas **Keys** stands for the encrypting keys, which are usually random variables in reality. But, we emphasize, that in the

formal view, these sets are purely sets of symbols. An example of an expression looks like

Example 1.2.

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4} \big), \Big(\big(K_2, \big\{ (\{001\}_{K_3}, \{K_6\}_{K_5}) \big\}_{K_5} \big), \{K_5\}_{K_2} \Big) \Big),$$

It is of course not necessary to use the first 7 keys, we could have used others. In our case,

$$\mathit{Keys}(M) = \{K_1, K_2, K_3, K_4, K_5, K_6, K_7\}.$$

□

Expressions are unambiguous, *i.e.,* $(M, N) = (M', N')$ means that $M = M'$ and $N = N'$, and $\{M\}_K = \{M'\}_{K'}$ means $M = M'$ and $K = K'$.

We will also need the notion of *subexpression* defined as follows:

Definition 1.3 (Sub-Expression)**.** We define the *set of sub-expressions* of an expression M, $\mathit{sub}(M)$, inductively as follows:

$$\begin{aligned}
&\mathit{sub}(K) = \{K\}, \text{ for } K \in \mathbf{Keys};\\
&\mathit{sub}(B) = \{B\}, \text{ for } B \in \mathbf{Blocks};\\
&\mathit{sub}((M, N)) = \mathit{sub}(M) \cup \mathit{sub}(N) \cup \{(M, N)\};\\
&\mathit{sub}(\{M\}_K) = \mathit{sub}(M) \cup \{\{M\}_K\}.
\end{aligned}$$

We say that N is a sub-expression of M, and denote by $N \sqsubseteq M$, if $N \in \mathit{sub}(M)$. □

In order to be able to formulate statements about expressions conveniently, we will need some more terminology, such as:

Definition 1.4 (Encrypted Expressions, Cyclic Set of Keys)**.** We say that a key K *encrypts* an expression N in M if there is an expression N', such that N is a subexpression of N' and $\{N'\}_K$ is a subexpression of M. For an expression M, we say that a subset $\mathcal{S}$ of $\mathit{Keys}(M)$ is *cyclic* in M if there exists $L_1, ..., L_n \in \mathcal{S}$ such that L_{i+1} encrypts L_i $(1 \leq i < n)$, and L_1 encrypts L_n, in M. □

Cyclicity of $Keys(M)$ can lead to serious difficulties in practice, and it is better to avoid them. Nevertheless, we try to reduce our conditions on cyclicity to the minimal, and in our theorems we will always indicate the cyclicity condition that is needed for the theorem to be true.

The problem of cyclicity suggests that it might be useful to restrict our attention to *valid expressions* $\mathbf{Exp}_{\mathcal{V}}$, which is a subset of $\mathbf{Exp}$ defined by some set of restricting rules (like cyclicity is excluded).

Definition 1.5 (Valid Expressions, Valid Ciphers)**.** A set of *valid expressions* is a subset $\mathbf{Exp}_{\mathcal{V}}$ of $\mathbf{Exp}$ such that

(i) all keys and all blocks are contained in $\mathbf{Exp}_{\mathcal{V}}$,

(ii) If $M \in \mathbf{Exp}_{\mathcal{V}}$ then all subexpressions of M are in $\mathbf{Exp}_{\mathcal{V}}$, and any number of pairs of these subexpressions are in $\mathbf{Exp}_{\mathcal{V}}$. Given a set of valid expressions, the set of *valid ciphers* is $\mathbf{Ciphers}_{\mathcal{V}} := \mathbf{Ciphers} \cap \mathbf{Exp}_{\mathcal{V}}$. □

1.1.2 What a formal adversary cannot distinguish: Equivalence

We continue the presentation of the Abadi-Rogaway logic by introducing the notion of *equivalence* of two expressions, which is meant to capture an adversary's incapability of distinguishing parts of expressions.

When an adversary looks at an expression built up from keys and blocks, he can see those keys that are not encrypted. With the help of these unencrypted keys, he can decrypt those messages that were encrypted by these keys, but he is unable to decrypt anything else. That is, when the adversary sees the message $(\{0\}_{K_{10}}, K_5)$, he cannot see what is inside the encrypted part, because K_{10} is not revealed, only K_5. On the other hand, if he sees $(\{0\}_{K_{10}}, K_{10})$, then he can decrypt the encryption, and hence see that there is a 0 inside. In our previous example, an adversary can see K_2, and with the help of that he can decrypt the last encryption (counting from left), hence revealing K_5; with the help of K_5, he can then reveal K_6, and see that the first encryption (from the left) contains 0. But, he cannot recover K_3 and K_4, and therefore, he cannot decrypt the parts that were encrypted with these keys.

Accordingly, we see that we need the notion of *recoverable* keys, those keys that can be extracted from an expression via a succession of descriptions. For the precise definition, we need

first a *key recovering function* $R : \mathbf{Exp} \times 2^{\mathbf{Keys}} \to 2^{\mathbf{Keys}}$, which, to a pair (M, S) of an expression M and a set of keys S, assigns that set of keys, which is the union of S and the keys that are recoverable from M with the help of the keys in S:

$R(K, S) := S \cup \{K\}$, for $K \in \mathbf{Keys}$;

$R(B, S) := S$, for $B \in \mathbf{Blocks}$;

$R((M, N), S) := R(M, S) \cup R(N, S)$;

$$R(\{M\}_K, S) := \begin{cases} S & \text{, if } K \notin S \\ R(M, S) & \text{, otherwise.} \end{cases}$$

With the help of this function, given an expression M, we can define a succession of sets of keys, $\mathfrak{K}_i(M)$, $(i = 0, 1, 2, ..)$ by

$\mathfrak{K}_0(M) = \emptyset$;

$\mathfrak{K}_i(M) = R(M, \mathfrak{K}_{i-1}(M))$.

Clearly,

$$\mathfrak{K}_0(M) \subseteq \mathfrak{K}_1(M) \subseteq \mathfrak{K}_2(M) \subseteq ...,$$

and, since an expression has finite length, there is an n natural, such that

$$\mathfrak{K}_n(M) = \mathfrak{K}_{n+1}(M) = \mathfrak{K}_{n+2}(M) = ...$$

Then,

Definition 1.6 (Recoverable Keys)**.**

$$\textit{R-Keys}(M) := \bigcup_i \mathfrak{K}_i(M).$$

□

Definition 1.7. For an expression M, let

$$\textit{B-Keys}(M) = \left\{ K \in \textit{Keys}(M) \;\middle|\; \begin{array}{l} \text{There is an } M' \text{ expression such that } \{M'\}_K \sqsubseteq M, \\ \text{and no unrecoverable key encrypts } \{M'\}_K \text{ in } M \end{array} \right\}.$$

□

We are now in a position to consider equivalence of expressions. Again, equivalence is supposed to express that two expressions look the same from an adversary's point of view. In the formalism that Abadi and Rogaway considered, to an adversary, all undecryptable encryptions look the same, therefore, equivalence is formulated via replacing undecryptable parts of a message by a box □. The formula that we obtain this way is called a *pattern*. Here is the formal definition:

Definition 1.8 (Pattern)**.** We define the *set of patterns*, $\mathbf{Pat}_0$, by the grammar:

$P, Q ::=$	*patterns*
K	key (for $K \in \mathbf{Keys}$)
B	block (for $B \in \mathbf{Blocks}$)
(P, Q)	pair
$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
$\Box$	undecryptable

□

We use the notation $\mathbf{Pat}_0$, because later we will define patterns that may contain more kinds of boxes.

We now define the function $patt_0$, which, given a set $S \subseteq \mathbf{Keys}$, creates a pattern from an expression by replacing all encryptions that cannot be decrypted by the keys in S by a box:

Definition 1.9. Let $S \subseteq \mathbf{Keys}$. We define the function $patt_0 : \mathbf{Exp} \times 2^{\mathbf{Keys}} \to \mathbf{Pat}_0$ inductively:

$$
\begin{aligned}
&patt_0(K, S) = K, && \text{for } K \in \mathbf{Keys} \\
&patt_0(B, S) = B, && \text{for } B \in \mathbf{Blocks} \\
&patt_0\big((M, N), S\big) = \big(patt_0(M, S), patt_0(N, S)\big) \\
&patt_0\big(\{M\}_K, S\big) = \begin{cases} \{patt_0(M, S)\}_K & \text{for } K \in S \\ \Box & \text{for } K \notin S \end{cases}
\end{aligned}
$$

□

Finally, *the pattern* of an expression is defined as

Definition 1.10 (The Pattern of an Expression)**.** For and expression M, let $pattern_0(M)$, be defined as

$$pattern_0(M) = patt_0(M, R\text{-}Keys(M)).$$

□

This exactly means that all encryptions in M that cannot be decrypted with keys recoverable from M itself, are replaced with a box.

Example 1.11. Continuing our example

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\big), \Big(\big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big),$$

The set of recoverable keys are

$$R\text{-}Keys(M) = \{K_2, K_5, K_6\},$$

and the pattern of M is

$$pattern_0(M) = \Big(\big(\{0\}_{K_6}, \ \square\ \big), \Big(\big(K_2, \big\{(\ \square\ , \{K_6\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big).$$

□

Now we are ready to define *equivalence* of two expressions, following the route taken by Abadi and Rogaway. In their treatment, elements in **Keys** model randomly generated keys that have the same distribution. That is, they all model the same key-generation algorithm. Therefore, replacing a key with another in an expression (each occurrence must be replaced), should result in an equivalent expression.

Definition 1.12 (Key-Renaming Function)**.** A bijection σ : **Keys** $\rightarrow$ **Keys** is called *key-renaming function*. For any expression (or pattern) M, $M\sigma$ denotes the expression (or pattern) obtained from M by replacing all keys K in M by $\sigma(K)$. □

Definition 1.13 (Equivalence of Expressions)**.** We say that two expressions M and N are *equivalent* (with respect to $pattern_0$), and denote it by

$$M \cong_0 N,$$

if there is a key-renaming σ such that

$$pattern_0(M)\sigma = pattern_0(N).$$

□

Example 1.14. Taking M to be the same as in our previous example, let

$$N = \Big(\big(\{0\}_{K_8}, \{100\}_{K_1}\big), \Big(\big(K_7, \big\{(\{0101\}_{K_9}, \{K_8\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_7}\Big)\Big).$$

Then

$$R\text{-}Keys(N) = \{K_5, K_7, K_8\},$$

and the pattern of N is

$$pattern_0(N) = \Big(\big(\{0\}_{K_8}, \ \square\ \big), \Big(\big(K_7, \big\{(\ \square\ , \{K_8\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_7}\Big)\Big).$$

Since, with the changes $K_2 \to K_7$, $K_5 \to K_5$ and $K_6 \to K_8$, the pattern of M turns into the pattern of N, M and N are equivalent.

On the other hand, for

$$N' = \Big(\big(\{0\}_{K_8}, \{100\}_{K_1}\big), \Big(\big(K_7, \big\{(\{0101\}_{K_9}, \{K_8\}_{K_5})\big\}_{K_7}\big), \{K_5\}_{K_7}\Big)\Big),$$

we have that

$$pattern_0(N') = \Big(\big(\{0\}_{K_8}, \ \square\ \big), \Big(\big(K_7, \big\{(\ \square\ , \{K_8\}_{K_5})\big\}_{K_7}\big), \{K_5\}_{K_7}\Big)\Big),$$

and therefore M and N' are not equivalent, because there is no bijection on the keys that transfers

the pattern of M into the pattern of N'. □

1.1.3 Expansions of the Abadi-Rogaway Formal Language

The equivalence of expressions as it was defined in the previous section is suitable for a situation when the formal language models protocols where an adversary cannot distinguish between any two encryptions with unknown encrypting keys. This, however, may not always be the case. For example, if, an adversary can detect whether two encryptions used the same keys or used different ones, then we cannot simply replace all encryptions with a single box, we need to differentiate the boxes according to the keys that were used for the encryption.

Another immediate concern is that a single key generation algorithm may not be enough, and we have to group the set of keys according to what kind of key generation processes they represent. For example, as we will see for the case of One-Time Pad, the length of the key needs to be matched with the length of the encrypted plaintext. Naturally, when equivalence is defined, in this case the key-renaming function σ should only be allowed to shuffle keys within the same group.

We therefore can expand the logic of Abadi and Rogaway in the following way: Let $\mathbf{Exp}_{\mathcal{V}}$ be a subset of valid expressions of **Exp**. We assume that an equivalence relation $\equiv_{\mathbf{K}}$ is given on the set of keys. For simplicity, we also assume, that each equivalence class contains infinitely many keys. Let

$$\mathcal{Q}_{\mathbf{Keys}} := \mathbf{Keys} \Big/ \equiv_{\mathbf{K}} .$$

Example 1.15. We will later see, that the formal language we have to use for One-Time Pad, includes a length-function $l : \mathbf{Keys} \to \{4, 5, ...\}$ on the keys (and ultimately on all valid expressions). Two keys there will be equivalent under $\equiv_{\mathbf{K}}$ if and only if they have the same length. □

We have to modify then the notion of key-renaming function, because keys should only be allowed to be renamed by other keys of the same sort. Therefore,

Definition 1.16 (Key-Renaming Function Relative to $\mathcal{Q}_{\mathbf{Keys}}$)**.** A bijection $\sigma : \mathbf{Keys} \to \mathbf{Keys}$ is called *key-renaming function*, if $\sigma(K) \equiv_{\mathbf{K}} K$ for all $K \in \mathbf{Keys}$. For any expression M, $M\sigma$

denotes the expression obtained by changing all keys in M to their images via σ. □

We now restrict the definition of valid expression, because we want to require that M is valid if and only if $M\sigma$ is valid:

Definition 1.17 (Valid Expressions, Valid Ciphers)**.** A set of *valid expressions* is a subset $\mathbf{Exp}_{\mathcal{V}}$ of **Exp** such that

(i) all keys and all blocks are contained in $\mathbf{Exp}_{\mathcal{V}}$,

(ii) if $M \in \mathbf{Exp}_{\mathcal{V}}$ then all subexpressions of M are in $\mathbf{Exp}_{\mathcal{V}}$, and any number of pairs of these subexpressions are in $\mathbf{Exp}_{\mathcal{V}}$,

(iii) for any σ key-renaming function relative to $\mathcal{Q}_{\mathbf{Keys}}$, $M \in \mathbf{Exp}_{\mathcal{V}}$ if and only if $M\sigma \in \mathbf{Exp}_{\mathcal{V}}$.

Given a set of valid expressions, the set of *valid ciphers* is $\mathbf{Ciphers}_{\mathcal{V}} := \mathbf{Ciphers} \cap \mathbf{Exp}_{\mathcal{V}}$. □

We also assume that there is an equivalence relation, $\equiv_{\mathbf{C}}$ given on the set of valid ciphers, with the property that form any $M, N \in \mathbf{Ciphers}_{\mathcal{V}}$ and σ is a key-renaming function relative to $\mathcal{Q}_{\mathbf{Keys}}$, $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$. The purpose of this relation is to capture what ciphers an adversary cannot distinguish, in other words, what partial information (length, key, etc...) can an adversary receive about the cipher. Let

$$\mathcal{Q}_{\mathbf{Ciphers}} := \mathbf{Ciphers}_{\mathcal{V}} \Big/ \equiv_{\mathbf{C}} .$$

Example 1.18. We will consider cryptosystems where an adversary can recognize when two ciphers were encrypted with different keys. For this case, we will need to define $\equiv_{\mathbf{C}}$ so that two ciphers are equivalent if and only if they are encrypted with the same key. □

Example 1.19. In [39], the authors find it useful to define a length-function on **Exp** in the

following way:

$$l(K) := 1 \quad \text{for } K \in \mathbf{Keys}$$
$$l(B) := 1 \quad \text{for } B \in \mathbf{Blocks}$$
$$l((M, N)) := l(M) + l(N)$$
$$l(\{M\}_K) := l(M) + 1$$

Two ciphers are then considered to be indistinguishable for an adversary if and only if they have the same length. Then, $\equiv_{\mathbf{C}}$ is chosen so that it equates ciphers with the same length, and an element of $\mathcal{Q}_{\mathbf{Ciphers}}$ will contain all ciphers that have a specific length. □

Naturally, we have to require (this is a condition on $\equiv_{\mathbf{C}}$), that renaming keys will carry equivalent ciphers into equivalent ones, *i.e.*, $M \equiv_{\mathbf{C}} N \in \mathbf{Ciphers}_{\mathcal{V}}$, if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$ whenever σ is a key-renaming function relative to $\mathcal{Q}_{\mathbf{Keys}}$. Hence a key-renaming function σ generates a renaming on $\mathcal{Q}_{\mathbf{Ciphers}}$, which we also denote by σ.

Definition 1.20 (A Formal Logic for Symmetric Encryption)**.** A formal logic for symmetric encryption is a triple $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ where $\mathbf{Exp}_{\mathcal{V}}$ is a set of valid expressions, $\equiv_{\mathbf{K}}$ is an equivalence relation on **Keys**, and $\equiv_{\mathbf{C}}$ is an equivalence relation on $\mathbf{Ciphers}_{\mathcal{V}}$; we require the elements of $\mathcal{Q}_{\mathbf{Keys}}$ to be infinite sets, and that for any σ key renaming function relative to $\mathcal{Q}_{\mathbf{Keys}}$,

(i) if $M \in \mathbf{Exp}$, then $M \in \mathbf{Exp}_{\mathcal{V}}$ if and only if $M\sigma \in \mathbf{Exp}_{\mathcal{V}}$,
(ii) if $M, N \in \mathbf{Ciphers}_{\mathcal{V}}$, then $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$. □

The patterns then need to reflect the fact that ciphers contained in different elements of $\mathcal{Q}_{\mathbf{Ciphers}}$ can be distinguished by an adversary. Therefore, we need to introduce a different box for each element $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$. Accordingly, the set $\mathbf{Pat}_{\Delta}$ is defined as

Definition 1.21 (Patterns for Δ)**.** We, again, proceed via induction:

$P, Q ::=$	*patterns*
K	key (for $K \in \mathbf{Keys}$)
B	block (for $B \in \mathbf{Blocks}$)
(P, Q)	pair
$\{P\}_K$	encryption by $K \in \mathbf{Keys}$
$\square_\mu$	undecryptable, with $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$.

□

The corresponding pattern-creating function is defined the following way:

Definition 1.22. Let $S \subseteq \mathbf{Keys}$. $patt_\Delta : \mathbf{Exp}_\mathcal{V} \times 2^{\mathbf{Keys}} \to \mathbf{Pat}_\Delta$ is defined inductively:

$$
\begin{aligned}
&patt_\Delta(K, S) = K, \qquad \text{for } K \in \mathbf{Keys}\\
&patt_\Delta(B, S) = B, \qquad \text{for } B \in \mathbf{Blocks}\\
&patt_\Delta\big((M, N), S\big) = \big(patt_\Delta(M, S), patt_\Delta(N, S)\big)\\
&patt_\Delta\big(\{M\}_K, S\big) = \begin{cases} \{patt_\Delta(M, S)\}_K & \text{if } K \in S\\ \square_\mu & \text{if } K \notin S \text{ and}\\ & \{M\}_K \in \mu,\ (\mu \in \mathcal{Q}_{\mathbf{Ciphers}}) \end{cases}
\end{aligned}
$$

□

Then

Definition 1.23 (The Pattern of a Valid Expression of Δ)**.** The pattern of a valid expression M, $pattern_\Delta(M)$, is

$$pattern_\Delta(M) = patt_\Delta(M, R\text{-}Keys(M)).$$

□

Example 1.24. In the case when the elements of $\mathcal{Q}_{\mathbf{Ciphers}}$ contain ciphers encrypted with the same key, there is a one-to-one correspondence between $\mathcal{Q}_{\mathbf{Ciphers}}$ and **Keys**, and therefore we

can index the boxes with keys instead of element in $\mathcal{Q}_{\mathbf{Ciphers}}$: $\square_K$, $K \in \mathbf{Keys}$. Then, if

$$N = \Big(\big(\{0\}_{K_8}, \{100\}_{K_1}\big), \Big(\big(K_7, \big\{(\{0101\}_{K_9}, \{K_8\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_7}\Big)\Big),$$

$$\textit{R-Keys}(N) = \{K_5, K_7, K_8\},$$

and the pattern of N is

$$\textit{pattern}_{\Delta}(N) = \Big(\big(\{0\}_{K_8}, \square_{K_1}\big), \Big(\big(K_7, \big\{(\square_{K_9}, \{K_8\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_7}\Big)\Big).$$

□

Finally,

Definition 1.25 (Equivalence of Valid Expressions of Δ)**.** We say that two valid expressions M and N are *equivalent*, and denote it by

$$M \cong_{\Delta} N,$$

if there is a key-renaming σ such that

$$\textit{pattern}_{\Delta}(M)\sigma = \textit{pattern}_{\Delta}(N),$$

where for any pattern Q, $Q\sigma$ denotes the pattern obtained by renaming the keys and the box-indexes in Q via σ. □

Example 1.26. Taking N as it was in our previous example, defining M as

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\big), \Big(\big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big),$$

The set of recoverable keys are

$$\textit{R-Keys}(M) = \{K_2, K_5, K_6\},$$

and the pattern of M is

$$pattern_{\Delta}(M) = \left(\left(\{0\}_{K_6},\ \Box_{K_1} \right), \left(\left(K_2, \left\{ (\ \Box_{K_9}\ , \{K_6\}_{K_5}) \right\}_{K_5} \right), \{K_5\}_{K_2} \right) \right).$$

Since, with the changes $K_2 \to K_7$, $K_5 \to K_5$, $K_6 \to K_8$, $K_4 \to K_1$ and $K_3 \to K_9$, the pattern of M turns into the pattern of N, M and N are equivalent. □

1.1.4 Proper Equivalence of Ciphers

In this section we introduce a natural property for the equivalence $\equiv_{\mathbf{C}}$, satisfied by a large class of expanded Abadi-Rogaway logics, namely all that is known to us with any significance.

Definition 1.27 (Proper Equivalence of Ciphers)**.** We say that an equivalence relation $\equiv_{\mathbf{C}}$ on $\mathbf{Ciphers}_{\mathcal{V}}$ is *proper*, if for any finite set of keys S, if $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$ contains an element of the form $\{N\}_K$ with $K \notin S$, then μ also contains an element C such that $Keys(C) \cap S = \emptyset$, and $K \not\sqsubseteq C$. □

In other words, if μ contains an element encrypted with a key K not in S, then μ has a representative in which no key of S appears, and in which K may only appear as an encrypting key, but not as a subexpression.

Example 1.28. If $\equiv_{\mathbf{C}}$ denotes the equivalence of example 1.18 (*i.e.* two ciphers are equivalent iff they have the same encrypting key), then it is clearly proper, since if $\{M\}_K \in \mu$, $K \notin S$, then $C = \{K'\}_K$ works for any $K' \notin S$; there is such a K', since we assumed that there are infinitely many keys. $C = \{B\}_K$ ($B \in \mathbf{Blocks}$) is also a good choice since **Blocks** is not empty. □

Example 1.29. If $\equiv_{\mathbf{C}}$ denotes the equivalence of example 1.19, then it is clearly proper, because if $\{M\}_K \in \mu$, $K \notin S$, then a good choice is $C = \{M'\}_K$ where M' is constructed from a $K' \notin S$ by pairing K' with itself $l(M)$ many times; there is such a K', since we assumed that there are infinitely many keys. □

The following propositions will be useful for proving our general soundness and completeness results.

To be able to state the following proposition, for each $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$, we first introduce the set

$$\mu_{\text{key}} := \left\{ K \in \mathbf{Keys} \;\middle|\; \text{there is an } M \text{ valid expression with } \{M\}_K \in \mu \right\}$$

Proposition 1.30. Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. Then, the equivalence relation $\equiv_{\mathbf{C}}$ is such that for any equivalence class $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$, μ_{key} has either one, or infinitely many elements.

Proof. Let $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$, and assume that there are more then one encrypting keys in μ, that is, there are two different keys K, K' such that $\{M\}_K, \{M'\}_{K'} \in \mu$ with some M, M' valid expressions. Since $\equiv_{\mathbf{C}}$ is proper, we can assume that $K \notin \mathit{Keys}(M')$. Since we assumed that each equivalence class in $\mathcal{Q}_{\mathbf{Keys}}$ contains infinitely many elements, there is a key L, such that $L \equiv_{\mathbf{K}} K$, and

$$L \notin \mathit{Keys}(\{M\}_K) \cup \mathit{Keys}(\{M'\}_{K'}).$$

Then, defining σ to do nothing else but to switch the keys L and K, $\{M\}_K\sigma = \{M\sigma\}_L$ and $\{M'\}_{K'}\sigma = \{M'\}_{K'}$. But, since

$$\{M\}_K \cong_{\mathcal{V}} \{M'\}_{K'},$$

it is also true that

$$\{M\}_K\sigma \cong_{\mathcal{V}} \{M'\}_{K'}\sigma.$$

Therefore, since $\{M'\}_{K'} \in \mu$, it must hold that $\{M\sigma\}_L \in \mu$. There are infinitely many choices for L, so there are infinitely many encrypting keys in μ. □

Proposition 1.31. Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. If σ is a key-renaming function (relative to $\equiv_{\mathbf{K}}$), then for any $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$, $|\mu_{\text{key}}| = |\sigma(\mu)_{\text{key}}|$.

Proof. If $|\mu_{\text{key}}| = \infty$, then $|\sigma(\mu)_{\text{key}}| = \infty$, since for any $\{M\}_K \in \mu$, $\{M\}_K\sigma = \{M\sigma\}_{\sigma(K)} \in \sigma(\mu)$. Since σ is a bijection, and since any μ contains either only one or infinitely many elements, the claim follows. □

The meaning of the next proposition is that if $\equiv_{\mathbf{C}}$ is proper, then given a set of valid ciphers $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ such that none of the encrypting keys are in S, and if $\mu_1, ...\mu_l$ are all the

equivalence classes of the elements in $\mathfrak{C}$, then it is possible to choose a representative of each of μ_j, denoted by C_{μ_j}, such that no key of S occurs in any of C_{μ_j}, none of L_i's occurs as a subexpression in any C_{μ_j}, and no key occurs in two of C_{μ_j} unless the corresponding two equivalence classes both have only the same, single encrypting key.

Proposition 1.32. Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. Let $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^{n}$ be a set of valid ciphers, and S a finite set of keys with $L_i \notin S$ ($i \in \{1, ..., n\}$). Let $\mu(\mathfrak{C})$ denote the set of all equivalence-classes with respect to $\equiv_{\mathbf{C}}$ of all elements in $\mathfrak{C}$. Then, for each $\nu \in \mu(\mathfrak{C})$, there is an element $C_\nu \in \nu$ such that

(i) $\mathit{Keys}(C_\nu) \cap S = \emptyset$ for all $\nu \in \mu(\mathfrak{C})$,

(ii) $L_i \not\sqsubseteq C_\nu$ for all $i \in \{1, ..., n\}$ and all $\nu \in \mu(\mathfrak{C})$,

(iii) and if $\nu \neq \nu'$, then $\mathit{Keys}(C_\nu) \cap \mathit{Keys}(C_{\nu'}) \neq \emptyset$ if and only if $\nu_{\text{key}} = \nu'_{\text{key}} = \{K\}$ (the set containing K only) for some K key, and in that case $\mathit{Keys}(C_\nu) \cap \mathit{Keys}(C_{\nu'}) = \{K\}$. Then, C_ν and $C_{\nu'}$ are both of the form $\{\cdot\}_K$ with the same K, and K is not a subexpression of either C_ν or $C_{\nu'}$.

Proof. Observe, that if $\mu(\{N_i\}_{L_i})$ denotes the equivalence class of $\{N_i\}_{L_i}$ in $\mathcal{Q}_{\mathbf{Ciphers}}$, then $\nu \in \mu(\mathfrak{C})$ if and only if $\nu = \mu(\{N_i\}_{L_i})$ for some $i \in \{1, ...n\}$.

Proof goes by induction. The statement is clearly true if $n = 1$, since $\equiv_{\mathbf{C}}$ is proper. Suppose it is true for $n - 1$. Let $\{N_1\}_{L_1}$, $\{N_2\}_{L_2}$,..., $\{N_n\}_{L_n}$ be valid expressions, and let S be a set of keys such that $L_i \notin S$. Without loss of generality, we can assume, that the numbering is such that there is an l, $1 \leq l \leq n$, such that

$$|\mu(\{N_n\}_{L_n})| = \begin{cases} 1 & \text{if } i \leq l \\ \infty & \text{if } i > l. \end{cases}$$

Let us first assume that $l = n$ and that there is an $m \in \{1, ..., n-1\}$ such that $L_m = L_n$. Since the statement is assumed to be true for $n - 1$, we can choose $C_{\mu(\{N_i\}_{L_i})}$ for $i \leq n - 1$ such that conditions (i), (ii), (iii) hold for these $\{C_{\mu(\{N_i\}_{L_i})}\}_{i=1}^{n-1}$ and S. If $\mu(\{N_n\}_{L_n}) = \mu(\{N_i\}_{L_i})$ for some $i \leq n-1$, then there is nothing to prove, $C_{\mu(\{N_n\}_{L_n})} = C_{\mu(\{N_i\}_{L_i})}$ has already been chosen.

If there is no such i, then consider

$$S_{n-1} := \left(\left(\bigcup_{i=1}^{n-1} \mathit{Keys}(C_{\mu(\{N_i\}_{L_i})}) \cup \{L_i\} \right) \setminus \{L_n\} \right) \cup S$$

According to the assumption, there is a $C \in \mu(\{N_n\}_{L_n})$ such that $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$ and $L_n \not\sqsubseteq C$, which implies that $C_{\mu(\{N_n\}_{L_n})} := C$ is a good choice: (i) follows from the fact that $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$. (ii) is true, since $L_m = L_n \not\sqsubseteq C$ is ensured by the way properness allows us to choose C; $L_i \not\sqsubseteq C$ for $i \leq n-1$, $i \neq m$, because $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$, and $L_n \not\sqsubseteq C_{\mu(\{N_i\}_{L_i})}$, because we assumed that $L_m = L_n$; $L_m \not\sqsubseteq C_{\mu(\{N_i\}_{L_i})}$ by the induction hypothesis. Finally, (iii) follows, because if $K' \in \mathit{Keys}(C_{\mu(\{N_i\}_{L_i})})$ for $i \in \{1, ..., n-1\}$, then, since $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$, $K' \in \mathit{Keys}(C)$ implies that $K' = L_n$. Clearly, in this case both $C_{\mu(\{N_i\}_{L_i})}$ and C are of the form $\{\cdot\}_{L_n}$, and L_n is not a subexpression of any of them because of the way properness allows us to chose the C_ν's.

If still $l = n$ holds, but there is no $m \in \{1, ..., n-1\}$ such that $L_m = L_n$, then $\{N_1\}_{L_1}$, $\{N_2\}_{L_2}$,..., $\{N_{n-1}\}_{L_{n-1}}$ and $S' := S \cup \{L_n\}$ satisfy the conditions of the induction hypothesis. If $l < n$, then $|\mu(\{N_n\}_{L_n})| = \infty$, and without loss of generality, we can assume that there is no $m \in \{1, ..., n-1\}$ such that $L_m = L_n$, and again, $\{N_1\}_{L_1}$, $\{N_2\}_{L_2}$,..., $\{N_{n-1}\}_{L_{n-1}}$ and $S' := S \cup \{L_n\}$ satisfy the conditions of the induction hypothesis. So in both of these cases, for S', we can choose $C_{\mu(\{N_i\}_{L_i})}$ for $i \leq n-1$ such that the conditions (i), (ii), (iii) hold for S' replacing S. If $\mu(\{N_n\}_{L_n}) = \mu(\{N_i\}_{L_i})$ for some $i \leq n$, then there is nothing to prove, $C_{\mu(\{N_n\}_{L_n})} = C_{\mu(\{N_i\}_{L_i})}$ has already been chosen; (i) and (iii) are obviously satisfied, and (ii) holds because $L_n \in S'$. If there is no such i, then consider

$$S_{n-1} := \left(\bigcup_{i=1}^{n-1} \mathit{Keys}(C_{\mu(\{N_i\}_{L_i})}) \cup \{L_i\} \right) \cup S.$$

According to the assumption, there is a $C \in \mu(\{N_n\}_{L_n})$ such that $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$, which implies that $C_{\mu(\{N_n\}_{L_n})} := C$ is a good choice: (i) follows from the fact that $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$. (ii) is true, since $L_n \not\sqsubseteq C$ because of properness; $L_i \not\sqsubseteq C$ for $i \leq n-1$ because $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$; $L_n \not\sqsubseteq C_{\mu(\{N_i\}_{L_i})}$, because $\mathit{Keys}(C_{\mu(\{N_i\}_{L_i})}) \cap (S \cup \{L_n\}) = \emptyset$ by the induction hypothesis. Finally, (iii) follows, because $\mathit{Keys}(C_{\mu(\{N_i\}_{L_i})}) \cap \mathit{Keys}(C_{\mu(\{N_n\}_{L_n})}) = \emptyset$ in this case. □

Given sets $\mathfrak{C}$ and S as in the conditions of the proposition, let $\mathfrak{R}(\mathfrak{C}, S)$ denote the nonempty set

$$\mathfrak{R}(\mathfrak{C}, S) := \left\{ \{C_\nu\}_{\nu \in \mu(\mathfrak{C})} \;\middle|\; \begin{array}{l} C_\nu \in \nu, \text{ and } \{C_\nu\}_{\nu \in \mathfrak{C}} \text{ and } S \text{ satisfy conditions} \\ \text{(i), (ii), (iii) of proposition 1.32} \end{array} \right\}$$

Another useful property satisfied by all common logics, and that we will need for the completeness result is the following:

Definition 1.33 (Independent $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$)**.** We say that $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent, if for any finite set of keys S, and any finite set $\mathfrak{C}$ of ciphers such that no key in S appears in any element of $\mathfrak{C}$, given any key-renaming function σ, there is a key renaming σ' for which $\sigma'(K) = K$ whenever $K \in S$, and for all $C \in \mathfrak{C}$, $C\sigma \equiv_{\mathbf{C}} C\sigma'$. □

In other words, $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent, if for any finite set of keys S, and any finite set $\mathfrak{C}$ of ciphers such that no key in S appears in any element of $\mathfrak{C}$, it is possible to alter any σ key-renaming function such that the altered key-renaming leaves all elements in S unchanged, whereas on $\mathfrak{C}$ it does the same thing as the original σ. We will need this property in for the general completeness theorem.

1.2 Computational Framework of Cryptographic Schemes

1.2.1 Symmetric Encryption Schemes

The computational modeling of encryption schemes provides a much more detailed description of a cryptographic protocol than the formal language that we presented in the previous section. It captures the fact that key generation and encryption is probabilistic, and it includes the fact that computers have limits in their computational power. Here, key generation algorithms are represented by random variables, messages are bit strings of finite length, and all algorithms, like encryption, decryption, key generation, must be computable in polynomial time relative to a so called "security parameter".

The field of actions here is the set $\mathbf{strings} := \{0,1\}^*$. A fixed subset, $\mathbf{plaintext} \subseteq \mathbf{strings}$ represents the messages that are allowed to be encrypted. We fix an element $\mathbf{0}$ in $\mathbf{plaintext}$. Another subset, $\mathbf{keys} \subseteq \mathbf{strings}$ is chosen for the possible encrypting keys. In order to be able to build up messages from basic ingredients, we assume that an injective *pairing function* is given:

$$[\ .\ ,\ .\ ] : \mathbf{strings} \times \mathbf{strings} \to \mathbf{strings}.$$

The range of the pairing function will be called **pairs**:

$$\mathbf{pairs} := \mathrm{Ran}_{[\ .\ ,\ .\ ]}.$$

A symmetric encryption scheme has the following constituents:

Security parameter. A security parameter η takes all values of the natural numbers. Computationally, it is represented by a finite string containing only 1, as many as its value is. The purpose of the security parameter is to measure the difficulty of computations. Functions, defined in terms of η, can be tested whether they are computable within a time interval that is no larger then some polynomial function of η.

Key-generation algorithm. Keys for encryptions are assumed to be randomly generated. The random generation must be computable in polynomial time with respect to the security parameter. Mathematically, key-generation is represented by a random variable $\mathcal{K}_\eta : \Omega_{K,\eta} \to \mathbf{keys}_\eta \subset \mathbf{keys}$, over a discrete probability field $(\Omega_{\mathcal{K},\eta}, \mathrm{Pr}_{\mathcal{K},\eta})$. We denote the function $\eta \mapsto \mathcal{K}_\eta$

by $\mathcal{K}$. As η in the index of the probability field indicates, the probability field can depend on the security parameter. We also put $\mathcal{K}$ in the index to remind that this probability field is for key generation, because we will consider other probability fields too.

Encryption algorithm. Encryption works as follows: For a given $k \in \mathbf{keys}$, and a given $x \in \mathbf{plaintext}$, $E_k(x)$, the encryption of x using k as the encrypting key is a random variable over some discrete probability field $(\Omega_E, \Pr_E)$. The values of this random variable are in **strings** and will be denoted by $E_k(x)(\omega)$, whenever $\omega \in \Omega_E$. We do not assume that $E_k(x)$ makes sense for any pair (k, x); we denote by Dom_E the subset of $\mathbf{keys} \times \mathbf{plaintext}$ for which $E_k(x)$ is defined. *I.e.,* if $\mathcal{RV}(\Omega_E, \mathbf{strings})$ stands for the set of random variables over Ω_E taking values in **strings**, then

$$E : \mathrm{Dom}_E \to \mathcal{RV}(\Omega_E, \mathbf{strings}),$$

where

$$\mathrm{Dom}_E \subseteq \mathbf{keys} \times \mathbf{plaintext}.$$

We assume that for a sequence of keys and x plaintext, $k_\eta \in \mathbf{keys}_\eta$, $E_{k_\eta}(x)$ is polynomial-time computable in η whenever it is defined. Let

$$\mathbf{ciphers} := \bigcup_{(k,x)\in \mathrm{Dom}_E} \mathrm{Ran}_{E_k(x)}$$

Decryption algorithm. An encryption must be decryptable, so we assume that for each $k \in \mathbf{keys}$, a function $D : (k, x) \mapsto D_k(x)$ is given on a subset Dom_D of $\mathbf{keys} \times \mathbf{strings}$ satisfying

$$D_k\big(E_k(x)(\omega)\big) = x$$

for all $\omega \in \Omega_E$. Again $D_{k_\eta}(x)$ must be polynomial-time computable.

Computational Equivalence In the computational setting, we assume that an adversary has access to computers with limited computing power. The purpose of security is that an adversary should have very small probability of getting valuable information about encrypted messages, which is expressed mathematically as having little chance to tell different ciphers from apart. Namely, messages are random variables, since key generation and encryption is random; more exactly, they are ensembles of random variables (because of the security parameter), and

the adversary is trying to distinguish these random ensembles. In order to express what it means to have little chance to distinguish two ensembles, we need the notion of *negligible function*:

Definition 1.34 (Negligible Function)**.** A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, if for any $c > 0$, there is an $n_c \in \mathbb{N}$ such that $\epsilon(\eta) \leq \eta^{-c}$ whenever $\eta \geq n_c$. □

Now we formulate when there is very little chance to tell two ensembles apart. Our definition is the same as the usual notion of computational indistinguishability, except that we formulate this for random variables that take values in a more general set then **strings**, because we will need that in our proofs. We consider random variables with values in a set, which is a Cartesian product of **strings** with itself finitely many times, that is, sets like $(\mathbf{strings} \times \mathbf{strings}) \times (\mathbf{strings} \times \mathbf{strings})$ (which we do not identify with $\mathbf{strings}^4$).

Definition 1.35 (Computationally Equivalent Probability Ensembles)**.** Let **str** be a set which is obtained by taking Cartesian products of **strings** with itself finitely many times. Let $F_\eta : \Omega_{F_\eta} \to \mathbf{str}$ and $G_\eta : \Omega_{G_\eta} \to \mathbf{str}$ be two sequences of random variables on $(\Omega_{F_\eta}, \Pr_{F_\eta})$ and $(\Omega_{G_\eta}, \Pr_{G_\eta})$ respectively. Let $\mathit{Dist}(F_\eta)$ and $\mathit{Dist}(G_\eta)$ denote their probability distributions. We say that the ensembles F_η and G_η (or, also, that $\mathit{Dist}(F_\eta)$ and $\mathit{Dist}(G_\eta)$) are *computationally equivalent*, if for any probabilistic polynomial-time adversary $\mathcal{A}_\eta$, *i.e.* for any function

$$\mathcal{A}_\eta : \Omega_{\mathcal{A}_\eta} \times \mathbf{str} \to \{0, 1\}$$

computable in polynomial time (where $(\Omega_{\mathcal{A}_\eta}, \Pr_{\mathcal{A}_\eta})$ is a probability field),

$$\begin{aligned} &\Pr_{\mathcal{A}_\eta} \otimes \Pr_{F_\eta} \Big(\big\{ (\omega_{\mathcal{A}}, \omega_{F_\eta}) \mid \mathcal{A}_\eta\big(\omega_{\mathcal{A}}, F_\eta(\omega_{F_\eta})\big) = 1 \big\} \Big) - \\ &- \Pr_{\mathcal{A}_\eta} \otimes \Pr_{G_\eta} \Big(\big\{ (\omega_{\mathcal{A}}, \omega_{G_\eta}) \mid \mathcal{A}_\eta\big(\omega_{\mathcal{A}}, G_\eta(\omega_{G_\eta})\big) = 1 \big\} \Big) \end{aligned}$$

is a negligible function of η. The tensor product stands for the product probability. □

Definition 1.36 (Encryption scheme)**.** A computational encryption scheme is a quadruple

$$\Pi = (\mathcal{K}, E, D, \approx)$$

where $\mathcal{K}$ is a key-generation algorithm, E is an encryption algorithm, D is a decryption algorithm that decrypts ciphers encrypted by E, and $\approx$ is the relation of computational equivalence. We require that the probability distribution of $\mathcal{K}$ be distinguishable from any constant string, and also that the distribution of (k, k') be distinguishable from the distribution of (k, k) if k and k' are independently generated: $k, k' \xleftarrow{R} \mathcal{K}_\eta$. Furthermore, we require the following property as well:

$$\text{either } \mathrm{Ran}_{\mathcal{K}_\eta} \times \{x\} \subset \mathrm{Dom}(E) \text{ for all } \eta, \text{ or } (\mathrm{Ran}_{\mathcal{K}_\eta} \times \{x\}) \cap \mathrm{Dom}(E) = \emptyset \text{ for all } \eta$$

holds for all and $x \in \mathbf{plaintext}$. □

The condition on the domain is to express the requirement that if for some, η value of the security parameter, and $\omega \in \Omega_{\mathcal{K},\eta}$, $E_{\mathcal{K}_\eta(\omega)}(x)$ is defined, then it is defined for all η, and, for any $\omega' \in \Omega_{\mathcal{K},\eta}$, $E_{\mathcal{K}_\eta(\omega')}(x)$ must also be defined. In other words, if x can be encrypted with a certain outcome of a key generation algorithm, then it must be encryptable with all outcomes of this same algorithm for all values of the security parameter.

1.2.2 Levels of Security

Security of computational encryption schemes is formulated via the notion of *encryption oracles*. For any key $k \in \mathbf{keys}$, an encryption oracle $\mathcal{E}_k(\cdot)$ assigns to an element $x \in \mathbf{plaintext}^*$, $x = (x_1, x_2, ..., x_l)$ the encrypted $(E_k(x_1), E_k(x_2), ..., E_k(x_l))$, where if $(k, x_i) \notin \mathrm{Dom}_E$ for some i, then $E_k(x_i)$ is $E_k(\mathbf{0})$. We will denote by $\mathcal{E}_k(\mathbf{0})$ the oracle that outputs $(E_k(\mathbf{0}), E_k(\mathbf{0}), ..., E_k(\mathbf{0}))$ upon receiving $x = (x_1, x_2, ..., x_l)$.

Definition 1.37 (Type-0 Security)**.** We say that a computational encryption scheme is type-0 secure, if no probabilistic polynomial-time adversary can distinguish the pair of oracles $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ and $(\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0}))$ as k and k' are randomly generated. That is, for any probabilistic polynomial-time algorithm, $\mathcal{A}_\eta$, querying either $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ or $(\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0}))$,

$$\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0})} = 1\right]$$

is a negligible function of η. □

The above formula means the following: The adversary is given one of two pairs of oracles, either $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ or $(\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0}))$ (where the keys were randomly generated prior to handing the pair to the adversary), but he does not know which. Then he has to query the pair and try to figure out which pair it is by flipping coins and doing polynomial-time computations with the responses from the oracles. The queries that the adversary submits may depend on the previous responses from the oracles. The keys that the oracles use for encryption do not change while the adversary queries the oracles. At the end, he has to come up with a number, either 1, or 0. Since the adversary is probabilistic, as well as the encryptions done by the oracles, moreover, the keys were randomly chosen at the beginning too, therefore, for each case, namely for each pair of oracles, there is a certain probability that the adversary comes up with 1 (or 0). We emphasize that this probability includes the randomness of the key generation done by the oracles at the beginning. If the difference of these two probabilities is negligible as a function of η, then we say that the two pairs are indistinguishable for the adversary, and hence the encryption is type-0 secure.

What type-0 security is meant to express is that not only no adversary can tell whether the oracles encrypt the plaintexts that the adversary submits or that they encrypt $\mathbf{0}$'s instead, but he cannot tell either whether the encryptions by the pair were done with the same key, or the keys had been separately generated.

If we do not require from the encryption scheme to be such that repetition of keys must not be detected, then the security level we get this way is called type-2 security:

Definition 1.38 (Type-2 Security)**.** We say that a computational encryption scheme is type-2 secure, if no probabilistic polynomial-time adversary can distinguish the oracles $\mathcal{E}_k(\cdot)$ and $\mathcal{E}_k(\mathbf{0})$ as k is randomly generated. That is, for any probabilistic polynomial-time algorithm, $\mathcal{A}_\eta$, querying either $\mathcal{E}_k(\cdot)$ or $\mathcal{E}_k(\mathbf{0})$,

$$\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\mathbf{0})} = 1\right]$$

is a negligible function of η. □

Another important notion in the literature is type-1 security, when key repetition is concealed, but the length of the encrypted message may be revealed. In this case, we require that adversaries

cannot distinguish pairs of oracles $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ and $(\mathcal{E}_k(0_{|\cdot|}), \mathcal{E}_k(0_{|\cdot|}))$, where $\mathcal{E}_k(0_{|\cdot|})$ denotes the oracle that upon receiving $x = (x_1, x_2, ..., x_l)$, returns

$$(E_k(0_{|x_1|}), E_k(0_{|x_2|}), ..., E_k(0_{|x_l|}))$$

where $E_k(0_{|x_i|})$ denotes encryption of the string of as many 0's as the length of x_i whenever $(k, x_i) \in \mathrm{Dom}_E$, and $E_k(\mathbf{0})$ if $(k, x_i) \notin \mathrm{Dom}_E$:

Definition 1.39 (Type-1 Security)**.** We say that a computational encryption scheme is type-1 secure, if no probabilistic polynomial-time adversary can distinguish the pair of oracles $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ and$(\mathcal{E}_k(0_{|\cdot|}), \mathcal{E}_k(0_{|\cdot|}))$ as k and k' are randomly generated. That is, for any probabilistic polynomial-time algorithm, $\mathcal{A}_\eta$, querying either $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ or $(\mathcal{E}_k(0_{|\cdot|}), \mathcal{E}_k(0_{|\cdot|}))$,

$$\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(0_{|\cdot|}), \mathcal{E}_k(0_{|\cdot|})} = 1\right]$$

is a negligible function of η. □

Finally, the combination of type-1 and type-2 security results in type-3 security, which might reveal both key repetition and length:

Definition 1.40 (Type-3 Security)**.** We say that a computational encryption scheme is type-3 secure, if no probabilistic polynomial-time adversary can distinguish the oracles $\mathcal{E}_k(\cdot)$ and $\mathcal{E}_k(0_{|\cdot|})$ as k is randomly generated. That is, for any probabilistic polynomial-time algorithm, $\mathcal{A}_\eta$, querying either $\mathcal{E}_k(\cdot)$ or $\mathcal{E}_k(0_{|\cdot|})$,

$$\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(0_{|\cdot|})} = 1\right]$$

is a negligible function of η. □

1.2.3 Hiding η

In this subsection we mention that it is possible to formulate the computational theory of cryptography by considering only single probability spaces instead of ensembles. The trick is, that

instead of considering random variables $f_\eta : \Omega_\eta \mapsto S$ over probability fields $(\Omega_\eta, \Pr_\eta)$, we can simply consider a single random variable over a field $(\Omega, \Pr)$:

$$F : \Omega \mapsto S^\infty,$$

with

$$F(\omega) = (F_1(\omega), F_2(\omega), ...)$$

such that the random variables $F_\eta : \Omega \mapsto S$ are all independent, and the distribution of F_η coincides with the distribution of f_η. The probability space Ω can be constructed via the infinite Cartesian product of the spaces Ω_η, whereas Pr is the product measure of all the $\Pr_\eta$.

In this terminology, instead of considering $\{0,1\}^*$ as the basic set on which encryption and decryption operates, we take $(\{0,1\}^*)^\infty$. Key generation results an element in $\mathbf{keys}^\infty$. The encryption E assigns to an element in $\mathbf{keys}^\infty \times \mathbf{plaintext}^\infty$ an element in $\mathbf{strings}^\infty$ by encrypting each component of the element in $\mathbf{plaintext}^\infty$ by the corresponding component of the element in $\mathbf{keys}^\infty$. Decryption is done similarly, component-wise.

Computational equivalence of two random variables of the form $F(\omega) = (F_1(\omega), F_2(\omega), ...)$ can of course be defined component-wise.

The advantage of this formalism is, that in this way we can incorporate the computational and the information-theoretic formalism that we present in the next section into a single formalism.

1.3 An Information-Theoretic Treatment

The information-theoretic treatment of cryptographic protocols is similar to the computational one, except that no condition on computational power is assumed, and hence no security parameter appears. We will only consider a specific example for this case, namely, the One-Time Pad (OTP), the idea of which originates from Shannon, who proved in [52] that for perfect secrecy, the encrypting keys has to be at least as long as the encrypted message is. One-Time Pad is the name for the encryption scheme that, given a plaintext with certain length, it generates a uniformly distributed random key with the same length as the plaintext, an the two are XOR'ed together. Each time, the key must be generated again to avoid leaking information. We will

consider a special implementation of OTP, that suits our purposes.

1.3.1 One-Time Pad

Consider the following realization of the one-time pad: Let

$$\textbf{strings} := \{0,1\}^*, \tag{1.1}$$

with the following pairing: For any two strings $x, y \in \textbf{strings}$ we can define the pairing as

$$[x,y] := \langle x, y, 0, 1_{|y|} \rangle \tag{1.2}$$

where $\langle\ ,\ ,\ \ldots\ ,\ \rangle$ denotes the concatenation of the strings separated by the commas, 1_m stands for m many 1's, and for any $x \in \{0,1\}^*$, $|x|$ denotes the length of the string. The number of 1's at the end indicate how long the second string is in the pair, and 0 separates the strings from the 1's.

Suppose N is a fixed natural number, and let

$$\textbf{blocks} := \{b | b \in \{0,1\}^*,\ b \text{ ends with } 100\} \tag{1.3}$$

The ending is just a tag, it shows that the meaning of the string is a block.

In case of the one-time pad, a plaintext is encrypted by generating a uniformly random binary string (a key) that has the same length as the plaintext and the two (the key and the plaintext) are XOR'ed together. Actually, we will need to tag the keys too, so, in fact we will generate a key that is three digits longer then the plaintext, but when we encrypt, we drop those last three digits. So let

$$\textbf{keys} := \{k \mid k \in \textbf{strings},\ k \text{ ends with } 010\}. \tag{1.4}$$

For any $n \in \mathbb{N}$, $n \geq 4$, let

$$\textbf{keys}_n := \{k \mid k \in \textbf{strings},\ |k| = n,\ k \text{ ends with } 010\}. \tag{1.5}$$

Key-Generation. For each key-length, there is a separate key generation algorithm, that is,

for each $n > 3$, $\mathcal{K}_n$ is a random variable over some $(\Omega_{\mathcal{K},n}, \Pr_{\mathcal{K},n})$ such that its values are equally distributed over $\mathbf{keys}_n$. For $k \in \mathbf{keys}$, let core(k) denote the string that we get from k by cutting the 010 at the end off.

Encryption. The domain of the encryption will be those elements $(k, x) \in \mathbf{keys} \times \mathbf{strings}$, for which $|k| = |x| + 3$:

$$\mathrm{Dom}_E := \{(k, x) \in \mathbf{keys} \times \mathbf{strings} \ \mid \ |k| = |x| + 3\} \tag{1.6}$$

This expresses that we can only use those keys for encryption that have the same length as the encrypted message. The encryption function is defined by

$$E(k, x) = E_k(x) := \langle \mathrm{core}(k) \oplus x, 110 \rangle \tag{1.7}$$

The tag 110 informs us, that the string is a cipher. Notice that this encryption is not probabilistic, $E_k(x)$ is not a random variable.

Let

$$\mathbf{ciphers} := \{c \mid c \in \mathbf{strings}, \ c \text{ ends with } 110\} \tag{1.8}$$

and

$$\mathbf{ciphers}_n := \{c \in \mathbf{ciphers} \mid |c| = n\}.$$

Observe, that **pairs**, **keys**, **blocks** and **ciphers** are all disjoint.

Decryption. The decryption function $D_k(x)$ is defined whenever $|k| = |x|$, and, naturally the value of $D_k(x)$ is the first $|k| - 3$ bits of $k \oplus x$.

Equivalence For One-Time Pad, we say that two probability distributions are equivalent if and only if they are identical.

Encryption Scheme Our realization of One-Time Pad is the quadruple

$$\Pi = (\{\mathcal{K}_i\}_{i \in \{4,5,\ldots\}}, E, D, \approx),$$

where the elements of the quadruple are the above defined set of key generations, encryption, decryption and equivalence.

1.4 A General Probabilistic Framework For Symmetric Encryption

As we have seen, it is possible to reformulate the computational view so that the probability fields and the random variables are not parametrized by the security parameter, instead it is already built into them. This fact allows us to give a formalism that includes both the computational and the information theoretic view. We list the mathematical objects of such a formalism, which are only slight modifications of what we have already seen in the computational and the information-theoretic framework.

The field of actions now is a more general set, $\overline{\textbf{strings}}$. In case of a computational framework, this is $(\{0,1\}^*)^\infty$, in the information-theoretic case, it is $\{0,1\}^*$. A fixed subset, $\overline{\textbf{plaintext}} \subseteq \overline{\textbf{strings}}$ represents the messages that are allowed to be encrypted. Another subset, $\overline{\textbf{keys}} \subseteq \overline{\textbf{strings}}$ is chosen for the possible encrypting keys. In order to be able to build up messages from basic ingredients, we assume that an injective *pairing function* is given:

$$[\ .\ ,\ .\ ] : \overline{\textbf{strings}} \times \overline{\textbf{strings}} \to \overline{\textbf{strings}}.$$

The range of the pairing function will be called $\overline{\textbf{pairs}}$:

$$\overline{\textbf{pairs}} := \mathrm{Ran}_{[\ .\ ,\ .\ ]}.$$

A symmetric encryption scheme has the following constituents:

Key-generation. Keys for encryptions are assumed to be randomly generated. Mathematically, key-generation is represented by a random variable $\mathcal{K} : \Omega_{\mathcal{K}} \to \textbf{keys}$, over a discrete probability field $(\Omega_{\mathcal{K}}, \Pr_{\mathcal{K}})$. We put $\mathcal{K}$ in the index to remind that this probability field is for key generation, because we will consider other probability fields too. In a given protocol, more then one key-generation is allowed.

Encryption. Encryption works as follows: For a given $k \in \overline{\textbf{keys}}$, and a given $x \in \overline{\textbf{plaintext}}$, $E_k(x)$ is a random variable over some discrete probability field $(\Omega_E, \Pr_E)$. The values of this random variable are in $\overline{\textbf{strings}}$ and are denoted by $E_k(x)(\omega)$, whenever $\omega \in \Omega_E$. We do not assume that $E_k(x)$ makes sense for any pair (k, x); we denote by Dom_E the subset of $\overline{\textbf{keys}} \times$

$\overline{\textbf{plaintext}}$ for which $E_k(x)$ is defined. *I.e.,* if $\mathcal{RV}(\Omega_E, \overline{\textbf{strings}})$ stands for the set of random variables over Ω_E taking values in $\overline{\textbf{strings}}$, then

$$E : \mathrm{Dom}_E \to \mathcal{RV}(\Omega_E, \overline{\textbf{strings}}),$$

where

$$\mathrm{Dom}_E \subseteq \overline{\textbf{keys}} \times \overline{\textbf{plaintext}}.$$

Decryption. An encryption must be decryptable, so we assume that for each $k \in \overline{\textbf{keys}}$, a function $D : (k, x) \mapsto D_k(x)$ is given on a subset Dom_D of $\overline{\textbf{keys}} \times \overline{\textbf{strings}}$ satisfying

$$D_k\big(E_k(x)(\omega)\big) = x$$

for all $\omega \in \Omega_E$.

Equivalence. Just as in the case of computational equivalence, we assume that an equivalence relation is defined on distributions over each finite Cartesian products of $\overline{\textbf{strings}}$ with itself. We call this notion *indistinguishability*, and denote it by $\approx$, no matter what the Cartesian product is, because this negligence will not lead to confusion. Let $\overline{\textbf{str}}$ be such a Cartesian product. We will also say that two random variables taking values in $\overline{\textbf{str}}$ are indistinguishable if (and only if) their distributions are equivalent (indistinguishable); we will use $\approx$ for denoting this equivalence between random variables as well. For $\approx$, we require the followings:

(i) Random variables with the same distribution are indistinguishable.

(ii) Constant random variables are indistinguishable if and only if the constants are the same.

For random variables F and G with values in $\overline{\textbf{str}}$, if $F \approx G$ then the followings must hold:

(iii) *Invariance under projections*: If $\pi^i_{\overline{\textbf{str}}}$ $(i = 1, 2)$ denotes the projection onto one of the two components of $\overline{\textbf{str}}$, then $\pi^i_{\overline{\textbf{str}}} \circ F \approx \pi^i_{\overline{\textbf{str}}} \circ G$.

(iv) *Invariance under coupling*: Let $\pi^i_{\overline{\textbf{str}}}$ be as above. If $F' : \Omega_F \to \overline{\textbf{str}}'$, $G' : \Omega_G \to \overline{\textbf{str}}'$ are also indistinguishable random variables such that F and F' are independent and G and G' are also independent, then $\omega_F \mapsto (F(\omega_F), F'(\omega_F))$ and $\omega_G \mapsto (G(\omega_G), G'(\omega_G))$ are indistinguishable random variables; moreover, if $\alpha, \beta : \overline{\textbf{str}} \to \overline{\textbf{str}}''$ are arbitrary functions that preserve $\approx$ (*i.e.* $\alpha \circ F \approx \alpha \circ G$ and $\beta \circ F \approx \beta \circ G$ whenever $F \approx G$), then $\omega_F \mapsto ((\alpha \circ F)(\omega_F), (\beta \circ F)(\omega_F))$ and

$\omega_G \mapsto ((\alpha \circ G)(\omega_G), (\beta \circ G)(\omega_G))$ are indistinguishable random variables if $F \approx G$.

(v) *Invariance under pairing*: If $\overline{\mathbf{str}} = \overline{\mathbf{strings}} \times \overline{\mathbf{strings}}$, then $[\cdot,\cdot] \circ F \approx [\cdot,\cdot] \circ G$.

(vi) *Invariance under depairing*: If $\overline{\mathbf{str}} = \mathbf{strings}$, then $[\cdot,\cdot]^{-1} \circ F \approx [\cdot,\cdot]^{-1} \circ G$.

We think of this relation as the generalized notion for computational or information-theoretic equivalence. It needs to satisfy some further properties under encryption and decryption that we specify under the definition of encryption schemes:

Definition 1.41 (Symmetric Encryption scheme)**.** A symmetric encryption scheme is a quadruple

$$\Pi = (\{\mathcal{K}^i\}_{i\in I}, E, D, \approx)$$

where $\{\mathcal{K}^i\}_{i\in I}$ is a set of key-generations for some index set I, E is an encryption, D decrypts ciphers encrypted by E and $\approx$ gives an equivalence relation as defined above. We require that for any $i \in I$, the probability distribution of $\mathcal{K}^i$ be distinguishable from any constant in $\overline{\mathbf{strings}}$, the distributions of $\mathcal{K}^i$ and of $\mathcal{K}^j$ be distinguishable whenever $i \neq j$, and also that the distribution of (k, k') be distinguishable from the distribution of (k, k) if k and k' are independently generated: $k \xleftarrow{R} \mathcal{K}^i$ $k' \xleftarrow{R} \mathcal{K}^j$ for any $i, j \in I$. Moreover, we require the following properties as well:

$$\text{either } \mathrm{Ran}_{\mathcal{K}^i} \times \{x\} \subset \mathrm{Dom}(E), \text{ or } (\mathrm{Ran}_{\mathcal{K}^i} \times \{x\}) \cap \mathrm{Dom}(E) = \emptyset$$

holds for all $i \in I$ and $x \in \overline{\mathbf{plaintext}}$. This property is to express the requirement that if for some $i \in I$, and $\omega \in \Omega_{\mathcal{K}}$, $E_{\mathcal{K}^i(\omega)}(x)$ is defined, then for all $\omega' \in \Omega_{\mathcal{K}}$, $E_{\mathcal{K}^i(\omega')}(x)$ must also be defined. In other words, if x can be encrypted with a certain outcome of a key generation algorithm, then it must be encryptable with all outcomes of this same algorithm for all values of the security parameter.

The equivalence relation $\approx$, besides satisfying the properties in the previous paragraph, needs to be such that if F and G are random variables taking values in $\overline{\mathbf{strings}}$, and $\mathcal{K}^i$ is key-generation such that the joint distribution of $(\mathcal{K}^i, F)$ is indistinguishable from the joint distribution of $(\mathcal{K}^i, G)$, then

(i) $(\omega_E, \omega_{\mathcal{K},i}, \omega) \mapsto E_{\mathcal{K}^i(\omega_{\mathcal{K},i})}\big(F(\omega)\big)(\omega_E)$ and $(\omega_E, \omega_{\mathcal{K},i}, \omega) \mapsto E_{\mathcal{K}^i(\omega_{\mathcal{K},i})}\big(G(\omega)\big)(\omega_E)$ are equivalent random variables.

(ii) $(\omega_{\mathcal{K},i}, \omega) \mapsto D_{\mathcal{K}^i(\omega_{\mathcal{K},i})}\big(F(\omega)\big)$ and $(\omega_{\mathcal{K},i}, \omega) \mapsto D_{\mathcal{K}^i(\omega_{\mathcal{K},i})}\big(G(\omega)\big)$ are also equivalent random

variables.

The probability over $\Omega_{\mathcal{K}^i} \times \Omega_F$ is the joint probability of $\mathcal{K}^i$ and F, which are here not necessarily independent. Similarly for G. The encryption is of course independent of $\mathcal{K}^i$ and F (and G).

Chapter 2

Interpretations and Soundness

In this chapter, we define the transition between the formal and the probabilistic views, called interpretation. Then we prove soundness theorems for the interpretations. The proves of the sondness theores are motivated by the soundness proof of Abadi and Rogaway in [3] where they used a hybrid argument as in [8, 21, 53]

2.1 Computational Interpretation of Formal Expressions

We would now like to relate the formal and the computational formalisms. For this purpose, we first consider a computational encryption scheme with only one key-generation algorithm: $\Pi = (\mathcal{K}, E, D, \approx)$. Naturally, what we want is to give a computational realization to the method of building messages via encryption and pairing. In order to make this work, we assume that $\mathbf{keys} \subseteq \mathbf{plaintext}$, $\mathbf{ciphers} \subseteq \mathbf{plaintext}$, $\mathbf{Blocks} \subseteq \mathbf{plaintext}$ and $[\mathbf{plaintext}, \mathbf{plaintext}] \subseteq \mathbf{plaintext}$ so that we can keep encrypting keys and ciphers.

The formal logic $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ suitable to be interpreted in computational encryption schemes is such that $\mathbf{Exp}_{\mathcal{V}} = \mathbf{Exp}$ (since there is no particular reason for restriction), and, since there is only one key generation algorithm, we will not distinguish formal keys either, so they are taken to be all equivalent. Hence, in this section we assume these from now on. We will talk about equivalence on formal ciphers later when we come to security assumptions. We assume that $\mathbf{0} \in B$.

Definition 2.1 (Interpretation for Computational Encryption Schemes)**.** Let $\Pi = (\mathcal{K}, E, D, \approx)$ be a computational symmetric encryption scheme with some index set I, with $(\Omega_{\mathcal{K},\eta}, \Pr_{\mathcal{K},\eta})$ denoting the probability fields for key generation, and with $(\Omega_E, \Pr_E)$ denoting the probability field for the randomness of encryption. For each valid expression $M \in \mathbf{Exp}_{\mathcal{V}} = \mathbf{Exp}$, let the probability space $(\Omega_{M,\eta}, \Pr_{M,\eta})$ be defined inductively as

$(\Omega_{K,\eta}, \Pr_{K,\eta}) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}})$ for $K \in \mathbf{Keys}$;

$(\Omega_{B,\eta}, \Pr_{B,\eta}) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}})$ for $B \in \mathbf{Blocks}$;

$(\Omega_{(M,N),\eta}, \Pr_{(M,N),\eta}) := (\Omega_{M,\eta} \times \Omega_{N,\eta}, \Pr_{M,\eta} \otimes \Pr_{N,\eta})$;

$(\Omega_{\{M\}_K,\eta}, \Pr_{\{M\}_K,\eta}) := (\Omega_E \times \Omega_{M,\eta}, \Pr_E \otimes \Pr_{M,\eta})$.

Where $(\{\omega_0\}, \mathbf{1}_{\{\omega_0\}})$ is just the trivial probability-space with one elementary event, ω_0 only; the tensor product stands for the product probability. Let

$$\Big(\Omega_{Keys(M),\eta}, \Pr_{Keys(M),\eta}\Big) := \Big(\times_{i=1}^{|Keys(M)|} \Omega_{\mathcal{K},\eta}, \otimes_{i=1}^{|Keys(M)|} \Pr_{\mathcal{K},\eta}\Big).$$

Let $\iota : Keys(M) \to \{1, .., |Keys(M)|\}$ a bijection enumerating the keys in $Keys(M)$. The function

$$(M, M') \mapsto \Big(\Phi_{M,\eta}(M') : \Omega_{M',\eta} \times \Omega_{Keys(M),\eta} \to \mathbf{strings}\Big)$$

defined whenever $M' \sqsubseteq M$, is *an interpreting function for computational encryption*, if it satisfies the following properties:

For $B \in \mathbf{Blocks}$, $B \sqsubseteq M$, $B \sqsubseteq N$,

$$\Phi_{M,\eta}(B)(\omega_0, \omega) = \Phi_{N,\eta}(B)(\omega_0, \omega') = B$$

for all M, N valid expressions and arbitrary $\omega \in \Omega_{Keys(M),\eta}$, $\omega' \in \Omega_{Keys(N),\eta}$.

For $K \in Keys(M)$,

$$\Phi_{M,\eta}(K)(\omega_0, (\omega_1, ..., \omega_{|Keys(M)|})) = \mathcal{K}_\eta(\omega_{\iota(K)}).$$

with $\omega_j \in \Omega_{\mathcal{K},\eta}$.

If $(M', M'') \sqsubseteq M$, then

$$\Phi_{M,\eta}((M', M''))((\omega', \omega''), \omega) = \big[\Phi_{M,\eta}(M')(\omega', \omega), \Phi_{M,\eta}(M'')(\omega'', \omega)\big].$$

for all $\omega' \in \Omega_{M',\eta}$, $\omega'' \in \Omega_{M'',\eta}$, and $\omega \in \Omega_{Keys(M),\eta}$.

Finally, if $\{M'\}_K \sqsubseteq M$, then

$$\Phi_{M,\eta}(\{M'\}_K)((\omega_E, \omega'), \omega) = E_{\Phi_{M,\eta}(K)(\omega_0,\omega)}\Big(\Phi_{M,\eta}(M')(\omega', \omega)\Big)(\omega_E) \tag{2.1}$$

for all $\omega_E \in \Omega_E$, $\omega' \in \Omega_{M',\eta}$, $\omega \in \Omega_{Keys(M),\eta}$. Let $\Phi_\eta(M) := \Phi_{M,\eta}(M)$. Let $[\![M]\!]_{\Phi_\eta}$ denote the distribution of $\Phi_\eta(M)$, and let $[\![M]\!]_\Phi$ denote the ensemble $\{[\![M]\!]_{\Phi_\eta}\}_{\eta\in\mathbb{N}}$. □

Clearly, equation 2.1 is not necessarily well-defined depending on what Dom_E is. We simply

assume, that Dom_E is such that this does not cause a problem, (another possibility is to restrict the set of valid expressions to those elements for which the interpretation is well-defined).

Intuitively, the random variable $\Phi_\eta(M)$ is the following. First, run key-generation for each key in $\mathit{Keys}(M)$. Then, using the outputs of these key-generations, translate the formal expressions according to the following rules: Each time you see a key, use the output of the corresponding key-generation. For blocks, just use the block itself. When you see a pairing, pair with $[\cdot,\cdot]$ the translations of the expressions inside the formal pair. When you see a formal encryption, run the encryption algorithm using the key string that was output by the key generation, encrypting the translation of the formal expression inside the formal encryption. The randomness of $\Phi_\eta(M)$ comes from the initial key-generation, and from running the encryption algorithm independently every time you encounter a formal encryption.

We also present a *conversion algorithm*, which was defined by Abadi and Rogaway in [3], and which results the same random variable as $\Phi_\eta(M)$ above:

Conversion Algorithm of Abadi and Rogaway. Let $\Pi = (\mathcal{K}, E, D, \approx)$ be a computational encryption scheme. The interpretation of an expression M using the encryption scheme Π can also be defined as follows:

algorithm INTERPRETATION(η, Q)
  for $K \in \mathit{Keys}(Q)$ **do** $\tau(K) \xleftarrow{R} \mathcal{K}_\eta$
  $y \xleftarrow{R}$ CONVERT(Q)
  return y

algorithm CONVERT(Q)
  if $Q = K$ where $K \in$ **Keys then**
    return $\tau(K)$
  if $Q = B$ where $B \in$ **Blocks then**
    return B
  if $Q = (Q_1, Q_2)$ **then**
    $x \xleftarrow{R}$ CONVERT(Q_1)
    $y \xleftarrow{R}$ CONVERT(Q_2)
    return $[x, y]$

if $Q = \{Q_1\}_K$ **then**

$x \xleftarrow{R} \mathrm{CONVERT}(Q_1)$

$y \xleftarrow{R} E_{\tau(K)}(x)$

return y

Then, the distribution of the sampling $y \xleftarrow{R} \mathrm{INTERPRETATION}(\eta, M)$ is identical with the distribution of $\Phi_\eta(M)$.

2.2 Soundness for Type-2 Encryption Scheme

We remind the reader to the definition of a type-2 security defined in section 1.2.2:

Definition 2.2 (Type-2 Security)**.** We say that an encryption scheme Π is type-2 secure if for any polynomial-time adversary A the following function is negligible as a function of η:

$$\mathrm{Adv}^2_{\Pi[\eta]}(A) := \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\mathbf{0})} = 1\right]$$

□

In a type-2 secure cryptosystem, an adversary may distinguish ciphers that were encrypted with different keys. Therefore, as we already mentioned in example 1.24, we should distinguish formal undecryptable ciphers as well, by assigning different boxes to them. Hence, the set of formal patterns should look like this:

Definition 2.3 (Pattern)**.** We define the *set of patterns*, $\mathbf{Pat}_2$, by the grammar:

$P, Q ::=$	*patterns*
K	key (for $K \in \mathbf{Keys}$)
B	block (for $B \in \mathbf{blocks}$)
(P, Q)	pair
$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
$\square_K$	undecryptable (for $K \in \mathbf{Keys}$)

□

Then, the process of assigning a pattern to an expression is the following:

Definition 2.4. Let $S \subseteq \mathbf{Keys}$. We define the function $patt_2 : \mathbf{Exp} \times 2^{\mathbf{Keys}} \to \mathbf{Pat}_2$ inductively as follows:

$$
\begin{aligned}
patt_2(K, S) &= K, \qquad \text{for } K \in \mathbf{Keys} \\
patt_2(B, S) &= B, \qquad \text{for } B \in \mathbf{blocks} \\
patt_2((M, N), S) &= (patt_2(M, S), patt_2(N, S)) \\
patt_2(\{M\}_K, S) &= \begin{cases} \{patt_2(M, S)\}_K & (\text{for } K \in S) \\ \square_K & (\text{for } K \notin S) \end{cases}
\end{aligned}
$$

□

Definition 2.5. We define the *pattern of an expression* M, $pattern_2(M)$, as

$$pattern_2(M) = patt_2(M, R\text{-}Keys(M))$$

□

We are now ready to state a lemma, and then the soundness theorem for this system:

Proposition 2.6. Consider an expression M, and a key $K_0 \in Keys(M)$. Suppose that for some expressions $M_1, M_2, ..., M_l \in \mathbf{Exp}$, $\{M_1\}_{K_0}, \{M_2\}_{K_0}, ..., \{M_l\}_{K_0}$ are subexpressions of M, and assume also that K_0 does not occur anywhere else in M. Then, denoting by M' the expression that we get from M by replacing each of $\{M_i\}_{K_0}$ that are not contained in any of M_j $(j \neq i)$ by $\{\mathbf{0}\}_{K_0}$, then following is true when Φ is an interpretation for a type-2 encryption scheme:

$$[\![M]\!]_\Phi \approx [\![M']\!]_\Phi. \tag{2.2}$$

Proof. We can assume, without loss of generality, that $\{M_i\}_{K_0}$ is a subexpression of $\{M_j\}_{K_0}$ only if $i < j$. Assume that (2.2) is not true. That is, suppose that $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$, which means that there is an adversary A that distinguishes the two distributions, that is

$$\Pr(x \xleftarrow{R} [\![M]\!]_{\Phi_\eta} : A_\eta(x) = 1) - \Pr(x \xleftarrow{R} [\![M']\!]_{\Phi_\eta} : A_\eta(x) = 1)$$

is a non-negligible function of η. We will show that this contradicts type-2 security. To this end, we construct an adversary that can distinguish between the oracles $\mathcal{E}_{k_0}(\cdot)$ and $\mathcal{E}_{k_0}(\mathbf{0})$. This adversary is the following probabilistic algorithm with f an oracle:

algorithm $A_\eta^f(M)$

for $K \in Keys(M) \setminus \{K_0\}$ **do** $\tau(K) \xleftarrow{R} \mathcal{K}_\eta$

$y \xleftarrow{R}$ CONVERT2(M)

$b \xleftarrow{R} A_\eta(y)$

return b

algorithm CONVERT2(N)

if $N = K$ where $K \in$ **Keys then**

return $\tau(K)$

if $N = B$ where $B \in$ **Blocks then**

return B

if $N = (N_1, N_2)$ **then**

$x \xleftarrow{R}$ CONVERT2(N_1)

$y \xleftarrow{R}$ CONVERT2(N_2)

return $[x, y]$

if $N = \{N_1\}_{K_0}$ **then**

$x \xleftarrow{R}$ CONVERT2(N_1)

$y \xleftarrow{R} f(x)$

return y

if $N = \{N_1\}_K$ $(K \neq K_0)$ **then**

$x \xleftarrow{R}$ CONVERT2(N_1)

$y \xleftarrow{R} E_{\tau(K)}(x)$

return y

Note that the algorithm CONVERT2 does almost the same as the algorithm CONVERT, except that while CONVERT carries out all necessary encryptions, CONVERT2 makes the oracles carry out the encryptions for K_0. Therefore, in the case, when the oracle f is $\mathcal{E}_{k_0}(\cdot)$, then CONVERT2(Q) will be a random sample from $[\![M]\!]_{\Phi_\eta}$, whereas if the oracle used is $\mathcal{E}_{k_0}(\mathbf{0})$, then

CONVERT2(Q) will be a random sample from $[\![M']\!]_{\Phi_\eta}$. Thus,

$$\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot)}(M) = 1\right] = \Pr(x \xleftarrow{R} [\![M]\!]_{\Phi_\eta} : A_\eta(x) = 1)$$

and

$$\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\mathbf{0})}(M) = 1\right] = \Pr(x \xleftarrow{R} [\![M']\!]_{\Phi_\eta} : A_\eta(x) = 1)$$

But, according to our assumption, $[\![M]\!]_\Phi$ and $[\![M']\!]_\Phi$ can be distinguished, that is,

$$\Pr(x \xleftarrow{R} [\![M]\!]_{\Phi_\eta} : A_\eta(x) = 1) - \Pr(x \xleftarrow{R} [\![M']\!]_{\Phi_\eta} : A_\eta(x) = 1)$$

is a non-negligible function of η and so, $\mathrm{Adv}^2_{\Pi[\eta]}(A)$ is also a non-negligible function of η. This implies that our scheme cannot be type-2 secure, which contradicts the assumption. Hence, we cannot have $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$. □

Theorem 2.7 (Soundness of Type-2 Schemes)**.** Let M and N be expressions such that *B-Keys*(M) and *B-Keys*(N) are not cyclic in M and N respectively. Let Π be a type-2 secure encryption scheme, Φ the interpretation of **Exp** for type-2 systems. Then,

$$M \cong_2 N \text{ implies } [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

Proof. Since $M \cong_2 N$ then, by definition, there exists a key-renaming bijection σ on **Keys** such that $\mathit{pattern}_2(M) = \mathit{pattern}_2(N\sigma)$. This means that the "boxes" occurring in $\mathit{pattern}_2(M)$ must occur in $\mathit{pattern}_2(N\sigma)$ and vice-versa. That is,

$$\textit{B-Keys}(M) = \textit{B-Keys}(N\sigma) = \textit{B-Keys}(N)\sigma$$

On the other hand, the subexpressions of $\mathit{pattern}_2(M)$ and of $\mathit{pattern}_2(N\sigma)$ outside the boxes must agree as well. Hence,

$$\textit{R-Keys}(M) = \textit{R-Keys}(N\sigma) = \textit{R-Keys}(N)\sigma.$$

Let $L_1, L_2, \ldots, L_b$ denote the keys in *B-Keys*(M) = *B-Keys*(N)σ. *B-Keys*(M) and *B-Keys*(N)

(and therefore *B-Keys*($N\sigma$) as well) are not cyclic by hypothesis, so without loss of generality, we can assume that the L_i's are numbered in such a way that L_i encrypts L_j only if $i < j$ (for a more detailed argument, see [3]; this means that those keys in *B-Keys*(M) that are deeper in M have a higher number).

Let $M_0 = M$. Let M_1 be the expression obtained from M_0 by replacing all subexpressions in M_0 of the form $\{\cdot\}_{L_1}$ by $\{0\}_{L_1}$. Let then M_i, $i \geq 2$, be the expression obtained from M_{i-1} by replacing all subexpressions in M_{i-1} of the form $\{\cdot\}_{L_i}$ by $\{\mathbf{0}\}_{L_i}$. We do this for all $i \leq b$ and it is easy to see that in M_b, replacing the subexpressions of the form $\{\mathbf{0}\}_{L_i}$ by $\Box_{L_i}$, we arrive at *pattern*$_2(M)$.

Note that in M_{i-1}, L_i can only occur as an encrypting key. The reason for this is that if L_i is a subexpression of M, then it has to be encrypted with some non-recoverable key, otherwise L_i would be recoverable; moreover, it has to be encrypted with some key in *B-Keys*(M) because a subexpression of M is either recoverable or ends up in a box when we construct *pattern*$_2(M)$. Now, the element in *B-Keys*(M) that encrypts L_i has to be an L_j with $j < i$. But, all subexpressions in M of the form $\{\cdot\}_{L_j}$ were already replaced by $\{\mathbf{0}\}_{L_j}$ when we constructed M_j, so L_i cannot appear in M_{i-1} in any other place than an encrypting key.

From Lemma 2.6, it follows that $[\![M_{i-1}]\!]_\Phi \approx [\![M_i]\!]_\Phi$, for all i, $1 \leq i \leq b$. Hence,

$$[\![M]\!]_\Phi = [\![M_0]\!]_\Phi \approx [\![M_b]\!]_\Phi. \tag{2.3}$$

Carrying out the same process for $N\sigma$ through $(N\sigma)_0$, $(N\sigma)_1$, ..., $(N\sigma)_b$ we arrive at

$$[\![(N\sigma)]\!]_\Phi = [\![(N\sigma)_0]\!]_\Phi \approx [\![(N\sigma)_b]\!]_\Phi. \tag{2.4}$$

Since we supposed that $M \cong_2 N$, that is, *pattern*$_2(M)$ = *pattern*$_2(N\sigma)$, and therefore M_b = *pattern*$_2(M)$ and $(N\sigma)_b$ = *pattern*$_2(N\sigma)$, we have

$$[\![M_b]\!]_\Phi = [\![(N\sigma)_b]\!]_\Phi. \tag{2.5}$$

Then, it is clearly true that

$$[\![N]\!]_\Phi = [\![N\sigma]\!]_\Phi \tag{2.6}$$

because permuting the keys in N does not have any effect in the distributions. Putting Equations (2.3), (2.4), (2.5) and (2.6) together the soundness result follows:

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

□

We illustrate via an example how the theorem works:

Example 2.8. Consider the expressions M and N as follows:

$$\begin{array}{cccccccccc} \{0\}_{K_6} & \{\{K_7\}_{K_1}\}_{K_4} & K_2 & \{\{0\}_{K_3}\}_{K_5} & \{K_6\}_{K_4} & \{111\}_{K_5} & 0 & \{K_1\}_{K_6} & \{K_5\}_{K_2} \\ \{K'_8\}_{K'_5} & \{K'_1\}_{K'_4} & K'_3 & \{\{1\}_{K'_6}\}_{K'_2} & \{K'_7\}_{K'_4} & \{111\}_{K'_2} & 0 & \{K'_1\}_{K'_5} & \{K'_2\}_{K'_3} \end{array}$$

We left out the parentheses for convenience, because they have no significance in the proof, they can be anywhere, but, of course, they must be in the same positions in M and N.

The respective 2-patterns are

$$\begin{array}{cccccccccc} \Box_{K_6} & \Box_{K_4} & K_2 & \{\Box_{K_3}\}_{K_5} & \Box_{K_4} & \{111\}_{K_5} & 0 & \Box_{K_6} & \{K_5\}_{K_2} \\ \Box_{K'_5} & \Box_{K'_4} & K'_3 & \{\Box_{K'_6}\}_{K'_2} & \Box_{K'_4} & \{111\}_{K'_2} & 0 & \Box_{K'_5} & \{K'_2\}_{K'_3} \end{array}$$

We see that $\mathit{pattern}_2(M) = \mathit{pattern}_2(N\sigma)$, i.e., $M \cong_2 N$, using the following substitution:

$$\begin{array}{lll} \sigma(K_2) = K'_3 & & \sigma(K_3) = K'_6 \\ \sigma(K_5) = K'_2 & & \sigma(K_4) = K'_4 \\ & & \sigma(K_6) = K'_5 \end{array}$$

$$\begin{array}{rcl} \mathit{Keys}(M) & = & \{K_1, K_2, \ldots, K_7\}; \\ \mathit{R\text{-}Keys}(M) & = & \{K_2, K_5\}; \\ \mathit{B\text{-}Keys}(M) & = & \{K_3, K_4, K_6\}. \end{array} \qquad \begin{array}{rcl} \mathit{Keys}(N) & = & \{K'_1, K'_2, \ldots, K'_8\}; \\ \mathit{R\text{-}Keys}(N) & = & \{K'_2, K'_3\}; \\ \mathit{B\text{-}Keys}(N) & = & \{K'_4, K'_5, K'_6\}. \end{array}$$

In the soundness theorem we mentioned that it is possible to assume without loss of generality that the keys in *B-Keys*(M) can be renamed $L_1, \ldots, L_b$ in such a way that L_i encrypts L_j only if $i < j$ (this means that those keys in *B-Keys*(M) that are deeper have a higher number). So, we rename in the following way:

$$\begin{aligned} L_1 &= K_3 = \sigma(K_3) = K'_6 \\ L_2 &= K_4 = \sigma(K_4) = K'_4 \\ L_3 &= K_6 = \sigma(K_6) = K'_5 \end{aligned}$$

We obtain the renamed expressions, M' and N':

$$\begin{array}{cccccccccc} \{0\}_{L_3} & \{\{K_7\}_{K_1}\}_{L_2} & K_2 & \{\{0\}_{L_1}\}_{K_5} & \{L_3\}_{L_2} & \{111\}_{K_5} & 0 & \{K_1\}_{L_3} & \{K_5\}_{K_2} \\ \{K'_8\}_{L_3} & \{K'_1\}_{L_2} & K'_3 & \{\{1\}_{L_1}\}_{K'_2} & \{K'_7\}_{L_2} & \{111\}_{K'_2} & 0 & \{K'_1\}_{L_3} & \{K'_2\}_{K'_3} \end{array}$$

Continuing with the argument we define M_0 as M' and M_i as the result of replacing all the expressions of the form $\{\cdot\}_{L_i}$ in M_{i-1} by $\{\mathbf{0}\}_{L_i}$, $1 \leq i \leq b$.

$$\begin{array}{rcccccccccc} M' = M_0: & \{0\}_{L_3} & \{\{K_7\}_{K_1}\}_{L_2} & K_2 & \{\{0\}_{L_1}\}_{K_5} & \{L_3\}_{L_2} & \{111\}_{K_5} & 0 & \{K_1\}_{L_3} & \{K_5\}_{K_2} \\ M_1 \quad : & \{0\}_{L_3} & \{\{K_7\}_{K_1}\}_{L_2} & K_2 & \{\{\mathbf{0}\}_{L_1}\}_{K_5} & \{L_3\}_{L_2} & \{111\}_{K_5} & 0 & \{K_1\}_{L_3} & \{K_5\}_{K_2} \\ M_2 \quad : & \{0\}_{L_3} & \{\mathbf{0}\}_{L_2} & K_2 & \{\{\mathbf{0}\}_{L_1}\}_{K_5} & \{\mathbf{0}\}_{L_2} & \{111\}_{K_5} & 0 & \{K_1\}_{L_3} & \{K_5\}_{K_2} \\ M_3 \quad : & \{\mathbf{0}\}_{L_3} & \{\mathbf{0}\}_{L_2} & K_2 & \{\{\mathbf{0}\}_{L_1}\}_{K_5} & \{\mathbf{0}\}_{L_2} & \{111\}_{K_5} & 0 & \{\mathbf{0}\}_{L_3} & \{K_5\}_{K_2} \\ \hline N_3 \quad : & \{\mathbf{0}\}_{L_3} & \{\mathbf{0}\}_{L_2} & K'_3 & \{\{\mathbf{0}\}_{L_1}\}_{K'_2} & \{\mathbf{0}\}_{L_2} & \{111\}_{K'_2} & 0 & \{\mathbf{0}\}_{L_3} & \{K'_2\}_{K'_3} \\ N_2 \quad : & \{K'_8\}_{L_3} & \{\mathbf{0}\}_{L_2} & K'_3 & \{\{\mathbf{0}\}_{L_1}\}_{K'_2} & \{\mathbf{0}\}_{L_2} & \{111\}_{K'_2} & 0 & \{K'_1\}_{L_3} & \{K'_2\}_{K'_3} \\ N_1 \quad : & \{K'_8\}_{L_3} & \{K'_1\}_{L_2} & K'_3 & \{\{\mathbf{0}\}_{L_1}\}_{K'_2} & \{K'_7\}_{L_2} & \{111\}_{K'_2} & 0 & \{K'_1\}_{L_3} & \{K'_2\}_{K'_3} \\ N' = N_0 : & \{K'_8\}_{L_3} & \{K'_1\}_{L_2} & K'_3 & \{\{1\}_{L_1}\}_{K'_2} & \{K'_7\}_{L_2} & \{111\}_{K'_2} & 0 & \{K'_1\}_{L_3} & \{K'_2\}_{K'_3} \end{array}$$

Then, clearly,

$$[\![M_3]\!]_\Phi = [\![N_3]\!]_\Phi,$$

because M_3 and N_3 are the same up to key-renaming. From lemma 2.6, it follows that

$$[\![M_0]\!]_\Phi \approx [\![M_1]\!]_\Phi \approx [\![M_2]\!]_\Phi \approx [\![M_3]\!]_\Phi,$$

and that

$$[\![N_0]\!]_\Phi \approx [\![N_1]\!]_\Phi \approx [\![N_2]\!]_\Phi \approx [\![N_3]\!]_\Phi.$$

Therefore,

$$[\![M']\!]_\Phi \approx [\![N']\!]_\Phi,$$

and hence

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

□

2.3 Interpretation of Expressions for One-Time Pad

Let $\Pi = (\{\mathcal{K}_n\}_{n=4}^{\infty}, E, D, \approx)$ be the realization of OTP that we discussed in section 1.3.1. In case of the OTP, lengths of the messages, and of the keys have vital importance. This notion though is not reflected in the formal view as we defined it in section 1.1.1. Therefore, we have to expand the logic so that we can talk about the length of an expression.

Definition 2.9 (Formal Length-Function for OTP)**.** We assume that some length function l : **Keys** $\rightarrow \{4, 5, ...\}$ is given of the keys symbols. The lengths of an expression is defined as $l(B) := |B| + 3$. We added 3 (to match the length of the tag). We define the length function on any expression in **Exp** by induction:

$$l((M, N)) := l(M) + 2l(N) + 1,$$

$$l(\{M\}_K) := \begin{cases} l(M) + 3 \text{ if } l(M) = l(K) - 3 \\ 0 \text{ otherwise} \end{cases}$$

□

The valid expressions than are defined as those expressions in which the length of the encrypted subexpressions match the length of the encrypting key, and, in which no key used twice to encrypt. This latter condition is necessary to prevent leaking information because of the properties of the OTP.

Definition 2.10 (Valid Formal Expressions For OTP)**.**

$$\mathbf{Exp}_{\mathrm{OTP}} := \left\{ M \in \mathbf{Exp} \;\middle|\; \begin{array}{l} M' \sqsubseteq M \text{ implies } l(M') > 0, \\ \text{and each key encrypts at most once in } M \end{array} \right\}.$$

□

An equivalence relation $\equiv_l$ is defined on the keys by $K_i \equiv_l K_j$ iff $l(K_i) = l(K_j)$. The equivalence classes contain keys of the same length, and they correspond to the different key-generation algorithms.

Definition 2.11 (Interpretation of an expression)**.** Given an expression $M \in \mathbf{Exp}_{\mathrm{OTP}}$, we create the interpretation in the following way. Suppose $\mathit{Keys}(M) = \{K_{i_1}, K_{i_2}, ..., K_{i_n}\}$. We first take

$$\Omega_{\mathit{Keys}(M)} := \times_{j=1}^{|\mathit{Keys}(M)|} \Omega_{\mathcal{K}, l(K_{i_j})},$$

the probability measure on this space is taken to be the product of the probabilities on the $\Omega_{\mathcal{K}, l(K_{i_j})}$. Let $\pi^j_{\Omega_{\mathit{Keys}(M)}}$ denote the projection onto the j'th component. We define

$$\Phi_M(K_{i_j}) : \Omega_{\mathit{Keys}(M)} \to \mathbf{keys},$$

$$\Phi_M(K_{i_j}) := \mathcal{K}_{l(K_{i_j})} \circ \pi^j_{\Omega_{\mathit{Keys}(M)}}.$$

Then, for each subexpression M' of M, we inductively define $\Phi_M(M')$ as a random variable $\Omega_{\mathit{Keys}(M)} \to \mathbf{strings}$:

For a block $B \in \mathbf{Blocks}$,

$$\Phi_M(B) : \Omega_{\mathit{Keys}(M)} \to \mathbf{blocks},$$

$$\Phi_M(B)(\omega) := \langle B, 100 \rangle.$$

For keys in $\mathit{Keys}(M)$, we already defined Φ_M.

For pairs of subexpressions of M,

$$\Phi_M((M', M''))(\omega) := \big[\Phi_M(M')(\omega), \Phi_M(M'')(\omega)\big].$$

Finally, for encryptions:

$$\Phi_M(\{M'\}_K)(\omega) := E_{\mathcal{K}_{l(K)}(\omega)}(\Phi_M(M')(\omega)).$$

Let $\Phi(M) = \Phi_M(M)$, and let $[\![M]\!]_\Phi$ denote the the distribution of $\Phi(M)$. □

We formulate this interpretation with the help of an algorithm as well:

algorithm INTERPRETATION$_{\mathrm{OTP}}(M)$
  for $K \in \mathit{Keys}(M)$ **do** $\tau(K) \xleftarrow{R} \mathcal{K}_{l(K)}$

$y \xleftarrow{R} \mathrm{CONVERT}_{\mathrm{OTP}}(M)$

return y

algorithm $\mathrm{CONVERT}_{\mathrm{OTP}}(N)$

if $N = K$ where $K \in$ **Keys then**

return $\tau(K)$

if $N = B$ where $B \in$ **Blocks then**

return $\langle B, 100 \rangle$

if $N = (N_1, N_2)$ **then**

return $[\mathrm{CONVERT}_{\mathrm{OTP}}(N_1), \mathrm{CONVERT}_{\mathrm{OTP}}(N_2)]$

if $N = \{N_1\}_K$ **then**

return $\langle E_{\tau(K)}(\mathrm{CONVERT}_{\mathrm{OTP}}(N_1)), 110 \rangle$

Then, the sampling $y \xleftarrow{R} \mathrm{INTERPRETATION}_{\mathrm{OTP}}(M)$ has the distribution $[\![M]\!]_{\Phi}$.

2.4 Soundness for One-Time Pad

As in the case of type-2 cryptosystems, we have to define equivalence of formal expressions suitable for OTP. This will mean differentiating boxes assigning different boxes two ciphers with different length. (That is, $\equiv_{\mathbf{C}}$ on ciphers is defined so that to ciphers are equivalent under $\equiv_{\mathbf{C}}$ iff they have the same length, just as in example 1.19.)

Definition 2.12 (Pattern)**.** We define the *set of patterns*, $\mathbf{Pat}_{\mathrm{OPT}}$, by the grammar:

$P, Q ::=$	*patterns*
K	key (for $K \in \mathbf{Keys}$)
B	block (for $B \in \mathbf{blocks}$)
(P, Q)	pair
$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
$\Box_n$	undecryptable (for $K \in \mathbb{N}$)

□

Then, the process of assigning a pattern to an expression is the following:

Definition 2.13. We define the function $patt_{\mathrm{OTP}} : \mathbf{Exp}_{\mathrm{OTP}} \times 2^{\mathbf{Keys}} \to \mathbf{Pat}_{\mathrm{OPT}}$ inductively as follows:

$$
\begin{aligned}
patt_{\mathrm{OTP}}(K, S) &= K, \qquad \text{for } K \in \mathbf{Keys} \\
patt_{\mathrm{OTP}}(B, S) &= B, \qquad \text{for } B \in \mathbf{blocks} \\
patt_{\mathrm{OTP}}((M, N), S) &= (patt_{\mathrm{OTP}}(M, S), patt_{\mathrm{OTP}}(N, S)) \\
patt_{\mathrm{OTP}}(\{M\}_K, S) &= \begin{cases} \{patt_{\mathrm{OTP}}(M, S)\}_K & (\text{for } K \in S) \\ \Box_{l(K)} & (\text{for } K \notin S) \end{cases}
\end{aligned}
$$

□

Definition 2.14. We define the *pattern of an expression* M, $pattern_{\mathrm{OTP}}(M)$, as

$$pattern_{\mathrm{OTP}}(M) = patt_{\mathrm{OTP}}(M, R\text{-}Keys(M))$$

□

Let then $\cong_{\mathrm{OPT}}$ denote the equivalence of expressions: Two expressions are equivalent if their patterns are the same up to a length-preserving key-renaming.

Again, we prove the soundness theorem via a lemma:

Proposition 2.15. Consider a valid expression $M \in \mathbf{Exp}_{\mathrm{OTP}}$, and a key $K_0 \in \mathit{Keys}(M)$. Suppose that for some expression M_0, $\{M_0\}_{K_0}$ is a subexpression of M, and assume also that K_0 does not occur anywhere else in M. Then, denoting by M' the expression that we get from M by replacing $\{M_0\}_{K_0}$ with $\{0_{l(K_0)-3}\}_{K_0}$ (where $0_{l(K_0)-3}$ denotes as string consisting of $l(K_0)-3$ many 0's), then following is true when Φ is the interpretation for OTP defined in section 2.3:

$$[\![M]\!]_\Phi = [\![M']\!]_\Phi. \tag{2.7}$$

Proof. The basic properties of the OTP ensure that $\Phi(\{M_0\}_{K_0})$ is evenly distributed over $\mathbf{ciphers}_{l(K_0)}$, no matter what M_0 is. So the distribution of $\Phi(\{M_0\}_{K_0}$ agrees with the distribution of $\Phi(\{0_{l(K_0)-3}\}_{K_0})$. Also, since K_0 is assumed not to occur anywhere else, $\Phi_M(K_0))$ is independent of the interpretation of the rest of the expression M, and therefore, $\Phi(\{M_0\}_{K_0})$ and $\Phi(\{0_{l(K_0)-3}\}_{K_0})$ are both independent of the interpretation of the rest of the expression. Hence, replacing $\Phi(\{M_0\}_{K_0})$ with $\Phi(\{0_{l(K_0)-3}\}_{K_0})$ will not effect the distribution. □

Theorem 2.16. Let M and N be two valid expressions in $\mathbf{Exp}_{\mathrm{OTP}}$ such that $\mathit{B\text{-}Keys}(M)$ and $\mathit{B\text{-}Keys}(N)$ are not cyclic in M and N respectively. Then $M \cong_{\mathrm{OTP}} N$ implies that $\Phi(M)$ and $\Phi(N)$ have the same probability distributions if Φ is the OTP interpretation.

Proof. Since $M \cong_{\mathrm{OTP}} N$ then, by definition, there exists a length-preserving bijection σ on **Keys** such that $\mathit{pattern}_{\mathrm{OTP}}(M) = \mathit{pattern}_{\mathrm{OTP}}(N\sigma)$. This means that the subexpressions of $\mathit{pattern}_{\mathrm{OTP}}(M)$ and of $\mathit{pattern}_{\mathrm{OTP}}(N\sigma)$ outside the boxes must agree. Hence,

$$\mathit{R\text{-}Keys}(M) = \mathit{R\text{-}Keys}(N\sigma)$$

Let $L_1, L_2, \ldots, L_b$ denote the keys in $\mathit{B\text{-}Keys}(M)$. Since $\mathit{B\text{-}Keys}(M)$ is not cyclic by hypothesis, we can assume that the L_i's are numbered in such a way that L_i encrypts L_j only if $i < j$.

Let $M_0 = M$. Let M_1 be the expression obtained from M_0 by replacing the subexpression in M_0 of the form $\{\cdot\}_{L_1}$ (there is only one such since encrypting twice with the same key is not permitted) by $\{0_{l(L_1)-3}\}_{L_1}$. Let then M_i, $i \geq 2$, be the pattern obtained from M_{i-1} by replacing that subexpression in M_{i-1} which has the form $\{\cdot\}_{L_i}$ with $\{0_{l(L_i)-3}\}_{L_i}$. We do this for all $i \leq b$. It is easy to see that in M_b, replacing the subexpressions of the form $\{0_{l(L_i)-3}\}_{L_i}$ by $\Box_{l(L_i)}$, we arrive at $\mathit{pattern}_2(M)$.

Note that in M_{i-1}, L_i can only occur as an encrypting key. The reason for this is that if L_i is a subexpression of M, then it has to be encrypted with some non-recoverable key, otherwise L_i would be recoverable (i.e., it would be visible); moreover, it has to be encrypted with some key in $\mathit{B\text{-}Keys}(M)$ because a subexpression of M is either visible or ends up in a box when we construct $\mathit{pattern}_{\mathrm{OTP}}(M)$. Now, the element in $\mathit{B\text{-}Keys}(M)$ that encrypts L_i has to be an L_j with $j < i$. But, all subexpressions in M of the form $\{\cdot\}_{L_j}$ were already replaced by $\{0_{l(L_j)-3}\}_{L_j}$ when we constructed M_j, so L_i cannot appear in M_{i-1} in any other place than an encrypting key.

From the previous proposition, it follows that $[\![M_{i-1}]\!]_\Phi = [\![M_i]\!]_\Phi$, for all i, $1 \leq i \leq b$. Hence,

$$[\![M]\!]_\Phi = [\![M_0]\!]_\Phi = [\![M_b]\!]_\Phi. \tag{2.8}$$

Carrying out the same process for $N\sigma$ through $(N\sigma)_0$, $(N\sigma)_1$, ..., $(N\sigma)_{b'}$, we arrive at

$$[\![(N\sigma)]\!]_\Phi = [\![(N\sigma)_0]\!]_\Phi = [\![(N\sigma)_{b'}]\!]_\Phi. \tag{2.9}$$

Now, since we supposed that $M \cong_{\mathrm{OTP}} N$, that is, $\mathit{pattern}_{\mathrm{OTP}}(M) = \mathit{pattern}_{\mathrm{OTP}}(N\sigma)$, and therefore M_b equals $(N\sigma)_{b'}$ up to a further key-renaming; we have

$$[\![M_b]\!]_\Phi = [\![(N\sigma)_{b'}]\!]_\Phi. \tag{2.10}$$

Then, it is clearly true that

$$[\![N]\!]_\Phi = [\![N\sigma]\!]_\Phi \tag{2.11}$$

because permuting the keys (with keeping the length) in N does not have any effect in the distributions. Putting Equations (2.8), (2.9), (2.10) and (2.11) together the soundness result

follows:

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

□

2.5 Interpretation and Soundness in General

2.5.1 Interpretation

The idea of the general interpretation is the same as what we had for the type-2 case and for the One-Time Pad. We do not give one specific interpretation though, we will just say that a function Φ is an interpretation if it satisfies certain properties. Here again, the interpretation $\Phi(M)$ of M is a random variable. We will assume, that a function ϕ is fixed in advance, that assigns to each formal key a key-generation algorithm. We will also assume that $\Phi(B)$ is given for blocks. Then, the rest of Φ is determined the following way: First, run the key-generation algorithm assigned by ϕ for each key in $\mathit{Keys}(M)$. Then, using the outputs of these key-generations, translate the formal expressions according to the following rules: Each time you see a key, use the output of the corresponding key-generation. For blocks, just use $\Phi(B)$. When you see a pairing, pair with $[\cdot,\cdot]$ the translations of the expressions inside the formal pair. When you see a formal encryption, run the encryption algorithm using the key string that was output by the key generation, encrypting the translation of the formal expression inside the formal encryption. The randomness of $\Phi(M)$ comes from the initial key-generation, and from running the encryption algorithm independently every time you encounter a formal encryption.

Definition 2.17 (Interpretation in the General Framework)**.** Let
$\Pi = (\{\mathcal{K}_i\}_{i\in I}^{\infty}, E, D, \approx)$ be a general symmetric encryption scheme with some index set I, with $\{(\Omega_{\mathcal{K}_i}, \Pr_{\mathcal{K}_i})\}_{i\in I}$ denoting the probability fields for key generation, and with $(\Omega_E, \Pr_E)$ denoting the probability field for the randomness of encryption. Let $\mathbf{Exp}_{\mathcal{V}}$ be a set of valid expressions. For each valid expression M, let the probability space $(\Omega_M, \Pr_M)$ be defined inductively as

$$(\Omega_K, \Pr_K) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \quad \text{for } K \in \mathbf{Keys};$$

$$(\Omega_B, \Pr_B) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \quad \text{for } B \in \mathbf{Blocks};$$

$$(\Omega_{(M,N)}, \Pr_{(M,N)}) := (\Omega_M \times \Omega_N, \Pr_M \otimes \Pr_N);$$

$$(\Omega_{\{M\}_K}, \Pr_{\{M\}_K}) := (\Omega_E \times \Omega_M, \Pr_E \otimes \Pr_M).$$

Where $(\{\omega_0\}, \mathbf{1}_{\{\omega_0\}})$ is just the trivial probability-space with one elementary event, ω_0 only; the tensor product stands for the product probability. Suppose that a function $\phi : \mathbf{Keys} \to \{\mathcal{K}_i\}_{i\in I}^{\infty}$

is given assigning key generations to abstract keys. Let $\iota : \{1, .., |Keys(M)|\} \to Keys(M)$ be a bijection enumerating the keys in $Keys(M)$. Let

$$\Big(\Omega_{Keys(M)}, \Pr_{Keys(M)}\Big) := \Big(\Omega_{\phi(\iota(1))} \times ... \times \Omega_{\phi(\iota(|Keys(M)|))}, \Pr_{\phi(\iota(1))} \otimes ... \otimes \Pr_{\phi(\iota(|Keys(M)|))}\Big).$$

The function

$$(M, M') \mapsto \Big(\Phi_M(M') : \Omega_{M'} \times \Omega_{Keys(M)} \to \overline{\mathbf{strings}}\Big)$$

defined whenever $M' \sqsubseteq M$, is called *an interpreting function*, if it satisfies the following properties:

For $B \in \mathbf{Blocks}$, $B \sqsubseteq M$, $B \sqsubseteq N$,

$$\Phi_M(B)(\omega_0, \omega) = \Phi_N(B)(\omega_0, \omega')$$

for all M, N valid expressions and arbitrary $\omega \in \Omega_{Keys(M)}$, $\omega' \in \Omega_{Keys(N)}$.

For $K \in Keys(M)$,

$$\Phi_M(K)(\omega_0, (\omega_1, ..., \omega_{|Keys(M)|})) = \phi(K)(\omega_{\iota^{-1}(K)}).$$

with $\omega_j \in \Omega_{\phi(\iota(j))}$.

If $(M', M'') \sqsubseteq M$, then

$$\Phi_M((M', M''))((\omega', \omega''), \omega) = \big[\Phi_M(M')(\omega', \omega), \Phi_M(M'')(\omega'', \omega)\big].$$

for all $\omega' \in \Omega_{M'}$, $\omega'' \in \Omega_{M''}$, and $\omega \in \Omega_{Keys(M)}$.

Finally, if $\{M'\}_K \sqsubseteq M$, then

$$\Phi_M(\{M'\}_K)((\omega_E, \omega'), \omega) = E_{\Phi_M(K)(\omega_0, \omega)}\Big(\Phi_M(M')(\omega', \omega)\Big)(\omega_E)$$

for all $\omega_E \in \Omega_E$, $\omega' \in \Omega_{M'}$, $\omega \in \Omega_{Keys(M)}$. Let $\Phi(M) := \Phi_M(M)$. □

Clearly, the definition is not necessarily well-defined depending on what Dom_E is. We simply assume, that Dom_E is such that this does not cause a problem, (another possibility is to restrict the set of valid expressions to those elements for which the interpretation is well-defined).

2.5.2 Soundness

The key to a soundness theorem is to have enough boxes in the definition of formal equivalence, *i.e.*, there should be enough elements in $\mathcal{Q}_{\mathbf{Ciphers}}$. It is clear that in the extreme case, when the equivalence on ciphers, $\equiv_{\mathbf{C}}$, is defined so that two ciphers are equivalent iff they are the same, then soundness holds trivially for all interpretations; but this would be completely impractical, it would assume a formal adversary that can see everything inside every encryption. It is also immediate, that if soundness holds with a given $\equiv_{\mathbf{C}}$ (and a given interpretation), and $\equiv'_{\mathbf{C}}$ is such that for any to formal ciphers M, N, $M \equiv'_{\mathbf{C}} N$ implies $M \equiv_{\mathbf{C}} N$ (i.e. $\equiv'_{\mathbf{C}}$ results more boxes), then, keeping the same interpretation, soundness holds with the new $\equiv'_{\mathbf{C}}$ as well. Hence, in a concrete situation, the aim is to introduce enough boxes to achieve soundness, but not too many, to sustain practicality. One way to avoid having too many boxes is to require completeness: we will see later, that obtaining completeness requires not to have too many boxes.

The following theorem claims the equivalence of two conditions. It is almost trivial that condition (i) implies condition (ii). The claim that (ii) implies (i) can be summarized the following way: if soundness holds for pairs of valid expressions M, M' with a special relation between them (described in (ii)), then soundness holds for all expressions (with certain acyclicity). In other words, if $M \cong_{\mathcal{V}} M'$ implies $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$ for certain specified pairs M, M', then $M \cong_{\mathcal{V}} N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ for any two pairs of valid expressions M, N.

For the definition of $\mathfrak{R}(\mathfrak{C}, S)$, see section 1.1.4.

Theorem 2.18. Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be a formal logic for symmetric encryption. Assume that $\equiv_{\mathbf{C}}$ is proper. Let $\Pi = (\{\mathcal{K}_i\}_{i\in I}^{\infty}, E, D, \approx)$ be a general encryption scheme, Φ an interpretation of $\mathbf{Exp}_{\mathcal{V}}$ in Π. Assume also, that for two keys $K, K' \in \mathbf{Keys}$ $K \equiv_{\mathbf{K}} K'$ if and only if $\phi(K) = \phi(K') = \mathcal{K}_i$ for some $i \in I$. Moreover, assume that replacing an undecryptable cipher within a valid expression with another equivalent undecryptable valid cipher results a valid expression. Then the following conditions are equivalent:

(i) Soundness holds for Φ: for any $M, N \in \mathbf{Exp}_{\mathcal{V}}$, if *B-Keys*$(M)$ and *B-Keys*(N) are not cyclic in M and N respectively, then $M \cong_{\Delta} N$, implies $\Phi(M) \approx \Phi(N)$.

(ii) For any $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^{n}$ set of valid ciphers, and S finite set of keys with $L_i \notin S$ $(i \in \{1, ..., n\})$, there is an element $\{C_\nu\}_{\nu\in\mu(\mathfrak{C})}$ of $\mathfrak{R}(\mathfrak{C}, S)$ such that the following holds: if $\left\{\{N_{i_j}\}_K\right\}_{j=1}^{l} \subset \mathfrak{C}$ and $M \in \mathbf{Exp}_{\mathcal{V}}$ are such that $\{N_{i_1}\}_K, \{N_{i_2}\}_K, ..., \{N_{i_l}\}_K \sqsubseteq M$, *R-Keys*$(M) \subset$

S, K does not occur anywhere else in M and *B-Keys*(M) is not cyclic in M, then if we denote by M' the expression obtained by replacing in M each of those $\{N_{i_j}\}_K$ that are not contained in any of $N_{i_{j'}}$ $(j \neq j')$ with $C_{\mu(\{N_{i_j}\}_K)}$, then

$$[\![M]\!]_\Phi \approx [\![M']\!]_\Phi. \tag{2.12}$$

Proof. Condition (ii) follows from (i) easily: For any set $\{C_{\mu(\{N_{i_j}\}_K)}\}_{i=1}^{l}$ provided by proposition 1.32, the encrypting key of $C_{\mu(\{N_{i_j}\}_K)}$ is not contained in S hence it is not recoverable key of M. Therefore, while computing the pattern of M', $C_{\mu(\{N_{i_j}\}_K)}$ will be replaced by the box $\square_{\mu(\{N_{i_j}\}_K)}$, which is the same box as the one that replaces $\{N_{i_j}\}_K$ in M when the pattern of M is computed. Hence $M \cong_\Delta M'$, and therefore, since soundness is assumed, and since *B-Keys*(M') is not cyclic in M',

$$[\![M]\!]_\Phi \approx [\![M']\!]_\Phi.$$

In order to prove that (i) follows from (ii), consider two equivalent valid expressions M and N, $M \cong_\Delta N$. Then, by definition, there exists a bijection σ on **Keys** (preserving $\equiv_{\mathbf{K}}$ such that $pattern_\Delta(M) = pattern_\Delta(N\sigma)$. This means that the "boxes" occurring in $pattern_\Delta(M)$ must occur in $pattern_\Delta(N\sigma)$ and vice-versa. Also, the subexpressions of $pattern_\Delta(M)$ and of $pattern_\Delta(N\sigma)$ outside the boxes must agree as well. Hence,

$$R\text{-}Keys(M) = R\text{-}Keys(N\sigma) = R\text{-}Keys(N)\sigma.$$

Let $L_1, L_2, \ldots, L_b$ $(L_i \neq L_j$ if $i \neq j)$ denote the keys in *B-Keys*(M), and let $L'_1, L'_2, \ldots, L'_{b'}$ $(L'_i \neq L'_j$ if $i \neq j)$ denote the keys in *B-Keys*$(N)\sigma$. *B-Keys*(M) and *B-Keys*(N) (and therefore *B-Keys*$(N\sigma)$ as well) are not cyclic by hypothesis, so without loss of generality, we can assume that the L_i's and the L'_i's are numbered in such a way that L_i encrypts L_j (and L'_i encrypts L'_j) only if $i < j$ (for a more detailed argument about this, see [3]; this means that those keys in *B-Keys*(M) that are deeper in M have a higher number).

Consider now the set of expressions that are subexpressions of M or N and have the form $\{M'\}_{L_i}$ or $\{N'\}_{L'_i}$, and also, the set S. Condition (ii) then provides the set with elements of the form $C_{\mu(\{M'\}_{L_i})}$ and $C_{\mu(\{N'\}_{L'_i})}$.

Let $M_0 = M$. Let M_1 be the expression obtained from M_0 by replacing all subexpressions in M_0 of the form $\{M'\}_{L_1}$ by $C_{\mu(\{M'\}_{L_1})}$ given by the assumption. Let then M_i, $i \geq 2$, be the expression obtained from M_{i-1} by replacing all subexpressions in M_{i-1} of the form $\{M'\}_{L_i}$ by $C_{\mu(\{M'\}_{L_i})}$. We do this for all $i \leq b$ and it is easy to see that in M_b replacing the subexpressions of the form $C_{\mu(\{M'\}_{L_i})}$ by $\Box_{\mu(\{M'\}_{L_i})}$ for all i, we arrive at $\mathit{pattern}_\Delta(M)$.

Note that in M_{i-1}, L_i can only occur as an encrypting key. The reason for this is that if L_i is a subexpression of M, then it has to be encrypted with some non-recoverable key, otherwise L_i would be recoverable; moreover, it has to be encrypted with some key in $\mathit{B\text{-}Keys}(M)$ because a subexpression of M is either recoverable or ends up in a box when we construct $\mathit{pattern}_\Delta(M)$. Now, the element in $\mathit{B\text{-}Keys}(M)$ that encrypts L_i has to be an L_j with $j < i$. But, all subexpressions in M of the form $\{M'\}_{L_j}$ were already replaced by $C_{\mu(\{M'\}_{L_j})}$ when we constructed M_j. According to the properties listed in proposition 1.32, L_i may only appear in $C_{\mu(\{M'\}_{L_j})}$ as the encrypting key, and then $L_i = L_j$, a contradiction. So L_i cannot appear in M_{i-1} in any other place than an encrypting key. Observe as well, that $\mathit{R\text{-}Keys}(M_i) = \mathit{R\text{-}Keys}(M)$.

From assumption (ii), it follows then that $[\![M_{i-1}]\!]_\Phi \approx [\![M_i]\!]_\Phi$, for all i, $1 \leq i \leq b$. Hence,

$$[\![M]\!]_\Phi = [\![M_0]\!]_\Phi \approx [\![M_b]\!]_\Phi. \tag{2.13}$$

Carrying out the same process for $N\sigma$ through $(N\sigma)_0$, $(N\sigma)_1$, ..., $(N\sigma)_b$ we arrive at

$$[\![(N\sigma)]\!]_\Phi = [\![(N\sigma)_0]\!]_\Phi \approx [\![(N\sigma)_b]\!]_\Phi. \tag{2.14}$$

Since we supposed that $M \cong_\Delta N$, that is, $\mathit{pattern}_\Delta(M) = \mathit{pattern}_\Delta(N\sigma)$, and therefore $M_b = \mathit{pattern}_\Delta(M)$ and $(N\sigma)_b = \mathit{pattern}_\Delta(N\sigma)$, we have

$$[\![M_b]\!]_\Phi = [\![(N\sigma)_b]\!]_\Phi. \tag{2.15}$$

Then, it is clearly true that

$$[\![N]\!]_\Phi = [\![N\sigma]\!]_\Phi \tag{2.16}$$

because permuting the keys in N does not have any effect in the distributions. Putting Equa-

tions (2.13), (2.14), (2.15) and (2.16) together the soundness result follows:

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

□

Example 2.19 (Type-2 Cryptosystems)**.** We can use this general theorem to give an alternative proof for the soundness for type 2 cryptosystems that was earlier discussed in section 2.2. We have to see that condition (ii) of theorem 2.18 is satisfied for this case. The equivalence relation $\equiv_{\mathbf{C}}$ in this case is proper as we mentioned in example 1.28. The equivalence relation $\equiv_{\mathbf{K}}$ is trivial here. The elements $\mu \in \mathcal{Q}_{\mathbf{Ciphers}}$ are in one-to-one correspondence with the keys, so we can say $\mathcal{Q}_{\mathbf{Ciphers}} \equiv \mathbf{Keys}$. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii), take $C_{L_i} := \{\mathbf{0}\}_{L_i}$. According to proposition 2.6, condition (ii) is satisfied. □

Example 2.20 (One-Time Pad)**.** The soundness theorem that we proved for One-Time Pad in section 2.4 is *not* an example for our general theorem. The reason is, that in condition (ii), we need a single C_ν for each $\nu \in \mu(\mathfrak{C})$ (*i.e.* for each length for OTP). On the other hand, in the case of the OTP, we cannot use the same encrypting key twice, therefore, we cannot use the same C_ν to replace equivalent ciphers within an expression. However, it is possible to define a formal equivalence and interpretation for One-Time Pad so that theorem 2.18 ensures soundness. The trick is to define $\equiv_{\mathbf{C}}$ so that two formal ciphers are equivalent, iff (again) the ciphers have the same encrypting key. The equivalence of keys, $\equiv_{\mathbf{K}}$ stays the same. Then, in this case too, the boxes will be indexed by the encrypting keys. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii), take $C_{L_i} := \{0_{l(L_i)-3}\}_{L_i}$. According to proposition 2.15, condition (ii) is satisfied. □

Example 2.21 (Type-1, Type-3 Cryptosystems)**.** For a discussion on type-1 systems, recall now example 1.19, where we cited the length-function Micciancio and Warinschi used in [39]. They assumed that the encryption scheme views the plaintext as a sequence of basic message blocks, and that a ciphertext is one block longer then the corresponding plaintext. (Practical encryption schemes such as CBC or CTR satisfy this property.) For the interpretation, they assumed that block symbols as well as key symbols are mapped to bit strings of size equal to one basic message

block. Therefore, the length-function is defined as

$$l(K) := 1 \quad \text{for } K \in \mathbf{Keys}$$

$$l(B) := 1 \quad \text{for } B \in \mathbf{Blocks}$$

$$l((M, N)) := l(M) + l(N)$$

$$l(\{M\}_K) := l(M) + 1$$

The equivalence of ciphers, $\equiv_{\mathbf{C}}$, for type-1 case is defined so that equivalence holds iff the formal length of the ciphers are the same. This gives a proper equivalence. A proof, analogous to that of proposition 2.6 shows that condition (ii) of our general theorem is satisfied.

It is clear that in order to be able to define equivalence on ciphers according to length, some length-function is needed to track the change in length via pairing and encrypting. This was easy in the previous example. However, in general, it is not necessarily true that a formal length-function can be defined. The problem is, that a length-function assigns a specific length to each expression, whereas an interpretation of an expression, which is a random variable, may have varying length. For example, in case of the One-Time Pad, the keys may be generated uniformly such that the length of the outcome of a key-generation varies (but, we have to require that the encrypting key is at least as long as the plaintext); the length of a cipher will also vary then.

For type-3 cryptosystems, equivalence on ciphers are defined so that equivalence holds iff the *encrypting keys and the lengths* agree. Then, again, a proposition similar to 2.6 shows that condition (ii) of the general theorem holds. □

Chapter 3

Completeness

Proving completeness of an interpretation of formal expressions means proving that whenever the interpretations of two expressions are indistinguishable, then the expressions themselves are equivalent. Clearly, this task can only be completed, if, having received a sample from an interpretation, we have a method to reveal all information contained in this sample string by revealing keys and decrypting ciphers. Hence, in section 3.1, we present a method of systematically revealing all possible information from a sample from the interpretation of a given expression.

3.1 Parsing Process

The technique that we present in this chapter will be very useful in the course of proving our completeness results. The idea can be summarized as follows: Given a sample element $x \xleftarrow{R} [\![M]\!]_\Phi$, x is built from blocks and randomly generated keys which are paired and encrypted. Some of the keys that were used for encryption when x was built might be explicitly contained in x, and in this case, using these keys, we can decrypt those ciphers that were encrypted with these revealed keys. The problem is though, that looking at x, it might not be possible to tell where blocks, keys, ciphers and pairs are in the string of bits, since we did not assume in general that we tag strings as we did for OTP. However, and we will exploit this fact repeatedly in our proofs, if we know that x was sampled from $[\![M]\!]_\Phi$ for a fixed, known M expression, then by looking at M, we can find in x the locations of blocks, keys, ciphers and pairs, and we can also tell from M, where the key decrypting a certain cipher is located. On the following couple of pages, we present a machinery that, using the form of an expression M, extracts from an $x \xleftarrow{R} [\![M]\!]_\Phi$ everything that is possible via decryption and depairing, and distributes the extracted elements over a special Cartesian product of copies of $\overline{\mathbf{strings}}$.

Throughout this section, we assume that $\Delta = (\mathbf{Exp}_\mathcal{V}, \equiv_\mathbf{K}, \equiv_\mathbf{C})$ and an interpretation Φ in a general symmetric encryption scheme $\Pi = (\{\mathcal{K}_i\}_{i\in I}, E, D, \approx)$ is given.

In this chapter we will often use the notion of *subexpression occurrence* of/in M. This means a subexpression together with its position in M. The reason for this distinction is that a subexpression can occur several times in M, and we want to distinguish these occurrences. But, to avoid cumbersome notation, we will denote the subexpression occurrence just as the subexpression itself. We start by defining the notion of 0-level subexpression occurrences of an expression M:

Definition 3.1 (Level 0 Subexpression Occurrences)**.** For an expression M, let us call *level 0 subexpression occurrences* all those subexpression occurrences in M that are not encrypted. Let $sub_0(M)$ denote the set of all level 0 subexpression occurrences in M. We write $N \sqsubseteq_0 M$ if N is a level 0 subexpression occurrence of N. □

Receiving an element $x \xleftarrow{R} [\![M]\!]_\Phi$, the first thing to do is to extract everything that is not encrypted, which means that we have to break up all pairs in x, and replace them with mathematical pairs. This process reveals the unencrypted blocks, keys and ciphers in x (i.e., the computational or statistical realizations of the 0-level subexpression occurrences).

Definition 3.2 (Blowup Function)**.** For each valid expression M, we define the *blowup function* $\mathcal{B}(M)$, on $\overline{\textbf{strings}}$ inductively as follows:

$$\mathcal{B}(K)x := x \quad \text{for } K \text{ key}$$
$$\mathcal{B}(B)x := x \quad \text{for } B \text{ block}$$
$$\mathcal{B}((M_1, M_2))x := (\mathcal{B}(M_1) \oplus \mathcal{B}(M_2)) \circ [\cdot, \cdot]^{-1}(x)$$
$$\mathcal{B}(\{N\}_K)x := x.$$

Where $\mathcal{B}(M_1) \oplus \mathcal{B}(M_2)$ denotes the function $(x, y) \mapsto (\mathcal{B}(M_1)x, \mathcal{B}(M_2)y)$. □

The element $\mathcal{B}(M)x$ is an element of $\mathcal{T}_0(M)$, which we define inductively the following way:

Definition 3.3 (Associated 0-Tree)**.** The *0-tree associated* to a pair of expressions N and M whenever $N \sqsubseteq_0 M$, will be denoted by $\mathcal{T}_0(N, M)$, and we define it as:

$$\mathcal{T}_0(K, M) := \overline{\textbf{strings}}$$
$$\mathcal{T}_0(B, M) := \overline{\textbf{strings}}$$
$$\mathcal{T}_0((M_1, M_2), M) := \mathcal{T}_0(M_1, M) \times \mathcal{T}_0(M_2, M)$$
$$\mathcal{T}_0(\{M'\}_K, M) := \overline{\textbf{strings}}$$

Let $\mathcal{T}_0(M) := \mathcal{T}_0(MM)$. □

We remind the reader that we do not identify $(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}) \times \overline{\mathbf{strings}}$ with $\overline{\mathbf{strings}} \times (\overline{\mathbf{strings}} \times \overline{\mathbf{strings}})$

Note also that for expressions $N \sqsubseteq_0 M'$ and $N \sqsubseteq_0 M$, we have that $\mathcal{T}_0(N, M') = \mathcal{T}_0(N, M)$. Nevertheless, we included M in the definition of $\mathcal{T}_0$ since for higher order trees, which we shall define later, the M in the second argument will make a difference.

Example 3.4. For the expression

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4} \big), \Big(\big(K_2, \big\{ (\{001\}_{K_3}, \{K_6\}_{K_5}) \big\}_{K_5} \big), \{K_5\}_{K_2} \Big) \Big),$$

$$sub_0(M) = \\ = \left\{ \begin{array}{l} \{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}, K_2, \big\{\{001\}_{K_3}, \{K_6\}_{K_5}\big)\big\}_{K_5}, \{K_5\}_{K_2}, \Big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4} \Big), \\ \Big(K_2, \big\{ (\{001\}_{K_3}, \{K_6\}_{K_5}) \big\}_{K_5} \Big), \Big(\Big(K_2, \big\{ (\{001\}_{K_3}, \{K_6\}_{K_5}) \big\}_{K_5} \Big), \{K_5\}_{K_2} \Big), M \end{array} \right\},$$

and

$$\mathcal{T}_0(M) = \Big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}} \Big) \times \Big((\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}) \times \overline{\mathbf{strings}} \Big).$$

Blocks, keys and ciphers are replaced by $\overline{\mathbf{strings}}$, pairs are replaced by $\times$. An element x sampled from $[\![M]\!]_\Phi$ looks like

$$\Big[\; [\; c_1 \; , \; c_2 \;] \; , \; \Big[\; [\; k \; , \; c_3 \;] \; , \; c_4 \; \Big] \; \Big]$$

where c_1 is a sample from $[\![\{0\}_{K_6}]\!]_\Phi$, c_2 is a sample from $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$, k is a sample from $[\![K_2]\!]_\Phi$, c_3 is a sample from $[\![\big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}]\!]_\Phi$, and c_4 is a sample from $[\![\{K_5\}_{K_2}]\!]_\Phi$. When we apply the blow-up function to this element x, we receive

$$\Big(\; (\; c_1 \; , \; c_2 \;) \; , \; \Big(\; (\; k \; , \; c_3 \;) \; , \; c_4 \; \Big) \; \Big)$$

which is an element of $\mathcal{T}_0(M)$. □

Proposition 3.5. For an expression M, if $x \xleftarrow{R} [\![M]\!]_\Phi$, then $\mathcal{B}(M)(x) \in \mathcal{T}_0(M)$.

Proof. Immediate from the definitions of $\mathcal{B}$ and $\mathcal{T}_0$. □

Perhaps it is even clearer if we label the copies of $\overline{\mathbf{strings}}$ in $\mathcal{T}_0(M)$ with the formal expressions that they belong to:

$$\begin{aligned}
\mathcal{T}_0'(K, M) &:= \overline{\mathbf{strings}}_K \\
\mathcal{T}_0'(B, M) &:= \overline{\mathbf{strings}}_B \\
\mathcal{T}_0'((M_1, M_2), M) &:= \mathcal{T}_0'(M_1, M) \times \mathcal{T}_0'(M_2, M) \\
\mathcal{T}_0'(\{M'\}_K, M) &:= \overline{\mathbf{strings}}_{\{M'\}_K}.
\end{aligned}$$

In our example,

$$\mathcal{T}_0'(M, M) = \left(\mathbf{s}_{\{0\}_{K_6}} \times \mathbf{s}_{\{\{K_7\}_{K_1}\}_{K_4}}\right) \times \left(\left(\mathbf{s}_{K_2} \times \mathbf{s}_{\{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}}\right) \times \mathbf{s}_{\{K_5\}_{K_2}}\right),$$

where we used $\mathbf{s}$ as a shorthand for $\overline{\mathbf{strings}}$.

In the previous example, c_4 is a random sample from $[\![\{K_5\}_{K_2}]\!]_\Phi$, and the function that projects onto the last copy of $\overline{\mathbf{strings}}$ in $\mathcal{T}_0(M)$, namely, onto $\overline{\mathbf{strings}}_{\{K_5\}_{K_2}}$, extracts c_4 from the blow-up. Similarly, projecting onto the other copies of $\overline{\mathbf{strings}}$, we extract samples form $[\![\{0\}_{K_6}]\!]_\Phi$, $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$ etc. To implement this idea in the general situation, we define what we can call the "0-Get Function" $\mathcal{G}_0(N, M)$ for an expression M and a subexpression occurrence N, whenever N is not encrypted in M. For $x \xleftarrow{R} [\![M]\!]_\Phi$, $\mathcal{G}_0(N, M)$ is to extract from $\mathcal{B}(M)x$ the sample of $[\![N]\!]_\Phi$ that was used for computing x. The precise definition goes by induction as usual:

Definition 3.6 (0-Get Function)**.** For subexpression occurrences $N \sqsubseteq_0 N' \sqsubseteq_0 M$, we define the *0-get function associated* to the triple (N, N', M), $\mathcal{G}_0(N, N', M) : \mathcal{T}_0(N', M) \to \mathcal{T}_0(N, M)$ inductively (the induction is for N') by

$$\mathcal{G}_0(N, N, M) := \mathrm{id}_{\mathcal{T}_0(N,M)}$$

$$\mathcal{G}_0(N, (M_1, M_2), M) := \begin{cases} \mathcal{G}_0(N, M_1, M) \circ \pi^1_{\mathcal{T}_0(M_1,M) \times \mathcal{T}_0(M_2,M)} & \text{if } N \text{ occurs in } M_1, \\ \mathcal{G}_0(N, M_2, M) \circ \pi^2_{\mathcal{T}_0(M_1,M) \times \mathcal{T}_0(M_2,M)} & \text{otherwise} \end{cases}$$

□

What we are really interested in is only $\mathcal{G}_0(N, M) := \mathcal{G}_0(N, M, M)$.

Example 3.7. In the previous example,

$$\mathcal{G}_0\big(\{0\}_{K_6}, M\big), \quad \mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big) : \mathcal{T}_0(M) \to \overline{\textbf{strings}}$$

$$\mathcal{G}_0\big(\{0\}_{K_6}, M\big)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = x_1,$$

$$\mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = x_2,$$

etc; that is, $\mathcal{G}_0\big(\{0\}_{K_6}, M\big)$ projects onto $\overline{\textbf{strings}}_{\{0\}_{K_6}}$, and $\mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big)$ projects onto $\overline{\textbf{strings}}_{\{\{K_7\}_{K_1}\}_{K_4}}$ etc. □

Observe, that for two expressions M and N, if $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, then for any $M' \in sub_0(M)$, there is a unique $N' \in sub_0(N)$ such that $\mathcal{G}_0(M', M) = \mathcal{G}_0(N', N)$. This motivates the following definition:

Definition 3.8 (Same Position of Subexpression Occurrences)**.** For two expressions M and N, if $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, we say that $M' \in sub_0(M)$ and $N' \in sub_0(M)$ *are in the same position at level 0*, if

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(N', N).$$

Let

$$\Gamma_0(N, M) : sub_0(M) \to sub_0(N)$$

denote the unique bijection such that

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(\Gamma_0(N, M)M', N)$$

for all $M' \in sub_0(M)$. □

Example 3.9. Let $N = ((0, 0), ((0, 0), 0)$. Then, if M denotes the expression from the previous examples, then $\mathcal{T}_0(N) = \mathcal{T}_0(M)$. Enumerating the 0's in N, we get the subexpression occurrences

$0_1 = 0$, $0_2 = 0$, $0_3 = 0$, $0_4 = 0$ and $0_5 = 0$, with $N = ((0_1, 0_2), ((0_3, 0_4), 0_5)$. Then

$$\Gamma_0(N, M)\{0\}_{K_6} = 0_1$$

$$\Gamma_0(N, M)\{\{K_7\}_{K_1}\}_{K_4} = 0_2$$

$$\Gamma_0(N, M)K_2 = 0_3$$

$$\Gamma_0(N, M)\big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5} = 0_4$$

$$\Gamma_0(N, M)\{K_5\}_{K_2} = 0_5$$

$$\Gamma_0(N, M)\Big(\{0\}_{K_6}, (\{\{K_7\}_{K_1}\}_{K_4}\Big) = (0_1, 0_2)$$

etc.

□

For an expression M, let $\mathcal{C}_M$ denote the set of all those subexpression occurrences in M which are ciphers encrypted by recoverable keys. *I.e.*,

$$\mathcal{C}_M = \{C \sqsubseteq M \mid C = \{M'\}_K \text{ for some } M' \sqsubseteq M \text{ and } K \in \textit{R-Keys}(M)\}.$$

We emphasize that we mean subexpression occurrences, that is, if an encryption encrypted with a recoverable key occurs twice in M, then it will be listed twice in $\mathcal{C}_M$. Since we assume that the elements of this set are encrypted by recoverable keys, it is possible to decrypt these elements one after the other, using only information containing M. Therefore, it is possible to enumerate the elements of this set in an order in which we can decrypt them by taking keys from M, decrypting what is possible with these keys and hence revealing more keys and then decrypting again with those keys etc. Let the total number of this set be denoted by $c(M)$. Then

$$\mathcal{C}_N = \{C^1, C^2, ..., C^{c(M)}\}.$$

Note that this enumeration is not unique. Also, note that the numbering does not mean that you can decrypt the ciphers only in this order. Let C^i_{key} denote the key that is used in the encryption C^i and let C^i_{text} denote the encrypted expression.

Example 3.10. In our example, the only possible way to enumerate is

$$C^1 = \{K_5\}_{K_2}$$
$$C^2 = \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}$$
$$C^3 = \{K_6\}_{K_5}$$
$$C^4 = \{0\}_{K_6}.$$

□

Now, to each expression M, we associate the "1-Decrypting Function" $\mathcal{D}_1(M)$. It acts on $\mathcal{T}_0(M)$. For an $x \in \mathcal{T}_0(M)$, $\mathcal{D}_1(M)$ takes $\mathcal{G}_0(C^1, M)x$ from $\overline{\mathbf{strings}}_{C^1}$, takes $\mathcal{G}_0(C^1_{\text{key}}, M)x$ from $\overline{\mathbf{strings}}_{C^1_{\text{key}}}$, and with the latter it decrypts the former if that is possible (namely, if they are of the right form: the former a cipher and the latter a key), and the result is broken into mathematical pairs, and what we get this way is put in the last component of the set $\overline{\mathbf{strings}} \times \{0\} \times \mathcal{T}_0(C^1_{\text{text}})$, and $\mathcal{G}_0(C^1_{\text{key}}, M)x$ goes into the first component. That is, the following element is created:

$$\Big(\ \mathcal{G}_0(C^1_{\text{key}}, M)x \ , \ 0 \ , \ \mathcal{B}(C^i_{\text{text}})\Big(\ D_{\mathcal{G}_0(C^1_{\text{key}}, M)x}\big(\mathcal{G}_0(C^1, M)x\big) \ \Big) \ \Big).$$

If $\Big(\mathcal{G}_0(C^1_{\text{key}}, M)x, \mathcal{G}_0(C^1, M)x\Big) \notin \text{Dom}_D$, then $\mathcal{D}_1(M)$ outputs $(0, 0, 0)$. The rest of $\mathcal{T}_0(M)$ is left untouched.

Let us introduce the notation

$$\mathcal{T}_0^{C^1}(M) = \big\{x \in \mathcal{T}_0(M) \ \big| \ \big(\mathcal{G}_0(C^1_{\text{key}}, M)x, \mathcal{G}_0(C^1, M)x\big) \in \text{Dom}_D \big\}.$$

Then,

Definition 3.11 (1-Decrypting Function)**.** For expressions $N \sqsubseteq M$, we define the map $\mathcal{D}_1(N, M)$

on $\mathcal{T}_0(M)$ inductively as follows: Let $x \in \mathcal{T}_0(M)$. Then

$$\mathcal{D}_1(K,M)x := \mathcal{G}_0(K,M)x$$

$$\mathcal{D}_1(B,M)x := \mathcal{G}_0(B,M)x$$

$$\mathcal{D}_1(\{M'\}_K,M)x := \mathcal{G}_0(\{M'\}_K,M)x \quad \text{if } K \notin \textit{R-Keys}(M)$$

$$\mathcal{D}_1((M_1,M_2),M)x := \big(\mathcal{D}_1(M_1,M)x\ ,\ \mathcal{D}_1(M_2,M)x\big)$$

$$\mathcal{D}_1(C^j,M)x :=$$

$$:= \begin{cases} \Big(\mathcal{G}_0(C^1_{\text{key}},M)x, 0, \mathcal{B}(C^1_{\text{text}})\big(D_{\mathcal{G}_0(C^1_{\text{key}},M)x}(\mathcal{G}_0(C^1,M)x)\big)\Big) \text{ if } x \in \mathcal{T}_0^{C^1}(M) \text{ and } j=1 \\ (0,0,0) \qquad \text{if } x \notin \mathcal{T}_0^{C^1}(M) \ \text{ and } \ j=1 \\ \mathcal{G}_0(C^j,M)x \qquad \text{if } j>1 \end{cases}$$

We introduce the notation $\mathcal{D}_1(M) := \mathcal{D}_1(M,M)$, this is what we will be interested in. □

We remark, that it is not important how we define $\mathcal{D}_1(C^1,M)x$ when $x \notin \mathcal{T}_0^{C^1}(M)$, we will not need that. We chose $(0,0,0)$ just for convenience.

Example 3.12. In the example that we have been using, that is, when

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\big), \Big(\big(K_2, \{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big),$$

with the choice $C^1 = \{K^5\}_{K_2}$,

$$\mathcal{D}_1(M)\big((x_1,x_2),((x_3,x_4),x_5)\big) = \begin{cases} \Big((x_1,x_2),\big((x_3,x_4),(x_3,0,\mathcal{B}(\{K_5\}_{K_2})(D_{x_3}(x_5)))\big)\Big) \text{ if} \\ \qquad (x_3,x_5) \in \text{Dom}_D \\ \big((x_1,x_2),((x_3,x_4),(0,0,0)\big) \quad \text{otherwise} \end{cases}$$

□

The target set of $\mathcal{D}_0(M)$ is naturally not $\mathcal{T}_0(M)$, because instead of the copy of $\overline{\textbf{strings}}$ corresponding to C_1 we now have a set of the form $\overline{\textbf{strings}} \times 0 \times \mathcal{T}_0(C^1_{\text{text}})$. We will call this new set $\mathcal{T}_1(M)$, and we right now extend the definition of $\mathcal{T}_0$ to higher order, up to $\mathcal{T}_{c(M)}(M)$. First we need the following following:

Definition 3.13 (Level i Subexpression Occurrences)**.** We will say that a subexpression occurrence $N \sqsubseteq M$ is level i with respect to $\mathcal{C}_M$, and denote this relation by $N \sqsubseteq_i M$, if the occurrence N is not in the occurrence C^j whenever $i < j$. Let $sub_i(M)$ denote the set of level i subexpression occurrences. □

Notice, that the level i subexpression occurrences are all those which are revealed once C^1, C^2, ... ,C^i are decrypted.

Definition 3.14 (Associated i-Tree)**.** We first inductively define the *i-tree associated* to a pair of expressions $N \sqsubseteq_i M$, and denote it by $\mathcal{T}_i(N, M)$:

$$\mathcal{T}_i(K, M) ::= \overline{\textbf{strings}}$$

$$\mathcal{T}_i(B, M) := \overline{\textbf{strings}}$$

$$\mathcal{T}_i((M_1, M_2), M) := \mathcal{T}_i(M_1, M) \times \mathcal{T}_i(M_2, M)$$

$$\mathcal{T}_i(C^j, M) := \begin{cases} \overline{\textbf{strings}} \times \{0\} \times \mathcal{T}_i(C^j_{\text{text}}, M) & \text{if } j \leq i \\ \overline{\textbf{strings}} & \text{otherwise} \end{cases}$$

$$\mathcal{T}_{i-1}(\{M'\}_K, M) := \overline{\textbf{strings}} \qquad \text{for } K \notin \textit{R-Keys}(M)$$

Let $\mathcal{T}_i(M) := \mathcal{T}_i(M, M)$. □

Note that we only "open" the encryptions that are made with the $\textit{R-Keys}(M)$ and at each step i we open only the C^j such that $j \leq i$.

Fact 3.15. For any expressions M and N, we have that $\mathcal{T}_i(M) \cap \mathcal{T}_i(N) = \emptyset$ or $\mathcal{T}_i(M) = \mathcal{T}_i(N)$. □

Similarly, we need to define $\mathcal{G}_i(N, M)$ and $\mathcal{D}_i(M)$ for $0 < i \leq c(M)$. The first one projects onto the copy of $\overline{\textbf{strings}}$ in $\mathcal{T}_i(M)$ that corresponds to N, and the second one maps an element in $\mathcal{T}_{i-1}(M)$ into $\mathcal{T}_i(M)$ decrypting the string corresponding to C^i with the appropriate key.

Definition 3.16 (i-Get Function)**.** For subexpression occurrences $N \sqsubseteq_i M$, $N' \sqsubseteq_i M$ ($0 \leq i \leq c(M)$) such that N occurs in N', we define the map *i-get-function associated* to the triple

(N, N', M), $\mathcal{G}_i(N, N', M) : \mathcal{T}_i(N', M) \to \mathcal{T}_i(N, M)$ inductively by:

$$\mathcal{G}_i(N, N, M) := \mathrm{id}_{\mathcal{T}_i(N,M)}$$

$$\mathcal{G}_i(N, (M_1, M_2), M) := \begin{cases} \mathcal{G}_i(N, M_1, M) \circ \pi^1_{\mathcal{T}_i(M_1,M)\times\mathcal{T}_i(M_2,M)} & \text{if } N \sqsubseteq M_1 \\ \mathcal{G}_i(N, M_2, M) \circ \pi^2_{\mathcal{T}_i(M_1,M)\times\mathcal{T}_i(M_2,M)} & \text{otherwise} \end{cases}$$

$$\mathcal{G}_i(N, C^j, M) := \mathcal{G}_i(N, C^j_{\text{text}}, M) \circ \pi^3_{\mathcal{T}_i(C^j_{\text{key}},M)\times\{0\}\times\mathcal{T}_i(C^j_{\text{text}},M)}, \text{ for } j \leq i, N \neq C^j$$

Define

$$\mathcal{G}_i(N, M) := \mathcal{G}_i(N, M, M).$$

□

Definition 3.17 (Same Position of Subexpression Occurrences)**.** For two expressions M and N, if $\mathcal{T}_i(M) = \mathcal{T}_i(N)$, we say that $M' \in sub_i(M)$ and $N' \in sub_i(M)$ *are in the same position at level* i, if

$$\mathcal{G}_i(M', M) = \mathcal{G}_i(N', N).$$

Let

$$\Gamma_i(N, M) : sub_i(M) \to sub_i(N)$$

denote the unique bijection such that

$$\mathcal{G}_i(M', M) = \mathcal{G}_i(\Gamma_i(N, M)M', N)$$

for all $M' \in sub_i(N)$. □

Let

$$\mathcal{T}^{C^i}_{i-1}(M) = \left\{ x \in \mathcal{T}_{i-1}(M) \;\middle|\; \left(\mathcal{G}_{i-1}(C^i_{\text{key}}, M)x, \mathcal{G}_{i-1}(C^i, M)x\right) \in \mathrm{Dom}_D \right\}.$$

Definition 3.18 (i-Decrypting Function)**.** For expressions $N \sqsubseteq_{i-1} M$ and $1 \leq i \leq c(M)$, we

define the map $\mathcal{D}_i(N,M) : \mathcal{T}_{i-1}(M) \to \mathcal{T}_i(N,M)$ inductively as follows: Let $x \in \mathcal{T}_{i-1}(M)$

$$\mathcal{D}_i(K,M)x := \mathcal{G}_{i-1}(K,M)x$$

$$\mathcal{D}_i(B,M)x := \mathcal{G}_{i-1}(B,M)x$$

$$\mathcal{D}_i(\{M'\}_K,M)x := \mathcal{G}_{i-1}(\{M'\}_K,M)x \quad \text{if } K \notin \textit{R-Keys}(M)$$

$$\mathcal{D}_i((M_1,M_2),M)x := (\mathcal{D}_i(M_1,M)x, \mathcal{D}_i(M_2,M)x)$$

$$\mathcal{D}_i(C^j,M)x :=$$

$$= \begin{cases} \Big(\mathcal{G}_{i-1}(C^j_{\text{key}},M)x, 0, \mathcal{D}_i(C^j_{\text{text}},M)\Big) & \text{if } j < i \\ \Big(\mathcal{G}_{i-1}(C^i_{\text{key}},M)x, 0, \mathcal{B}(C^i_{\text{text}})\big(D_{\mathcal{G}_{i-1}(C^i_{\text{key}},M)x}(\mathcal{G}_{i-1}(C^i,M)x)\big)\Big) & \text{if } x \in \mathcal{T}^{C^i}_{i-1}(M) \text{ and } j = i \\ (0,0,0) & \text{if } x \notin \mathcal{T}^{C^i}_{i-1}(M) \text{ and } j = i \\ \mathcal{G}_{i-1}(C^j,M)x & \text{if } j > i \end{cases}$$

Let

$$\mathcal{D}(M) := \mathcal{D}_{c(M)}(M) \circ ... \circ \mathcal{D}_1(M) \circ \mathcal{B}(M)$$

□

The functions $\mathcal{D}_i(M)$ one after the other decrypt all the ciphers that are encrypted with recoverable keys. Finally, $\mathcal{D}(M)$ decrypts all ciphers encrypted with recoverable keys upon an input from sampling $[\![M]\!]_\Phi$.

Example 3.19. In our on-going example,

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\big), \Big(\Big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\Big), \{K_5\}_{K_2}\Big)\Big),$$

If y is a sample from $[\![M]\!]_\Phi$, then $\mathcal{D}(M)y$ has the form

$$\Big(\big((y_6,0,0),y_1\big), \Big(\big(y_2, \big(y_5, 0, (y_3, (y_5, 0, y_6))\big)\big), \big(y_2, 0, y_5\big)\Big)\Big),$$

Where y_2, y_5, y_6 are outcomes of the key-generation algorithms $\phi(K_2)$, $\phi(K_5)$, $\phi(K_6)$ respectively, y_1 is an undecryptable sample element from $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$, and y_3 is an undecryptable sample from $[\![\{001\}_{K_3}]\!]_\Phi$. Moreover, $(y_6,0,0)$ indicates that the key y_6 encrypts the plaintext 0, $(y_2,0,y_5)$ indicates that the key y_2 encrypts the plaintext y_5 (which is also a key), and so on. □

The following lemma essentially claims that if the interpretation is such that conditions (i) and (ii) bellow hold, then for any two valid expressions, M and N, the distribution of $\mathcal{D}(M)x$, where x is sampled from $[\![M]\!]_\Phi$ (let $\mathcal{D}(M)([\![M]\!]_\Phi$ denote this distribution), is indistinguishable from the distribution of $\mathcal{D}(N)y$, where y is sampled from $[\![N]\!]_\Phi$ whenever $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$.

For a function f on $\overline{\mathbf{strings}}$, let $f([\![M]\!]_\Phi)$ denote the probability ditribution of $f(x)$ as x is sampled from $[\![M]\!]_\Phi$.

Lemma 3.20. Let $\Delta = (\mathbf{Exp}_\mathcal{V}, \equiv_\mathbf{K}, \equiv_\mathbf{C})$ be a formal logic for symmetric encryption, and let Φ be an interpretation of $\mathbf{Exp}_\mathcal{V}$ in $\Pi = (\{\mathcal{K}_i\}_{i\in I}, E, D, \approx)$. Suppose that this realization satisfies the following properties for any $K, K', K'' \in \mathbf{Keys}$, $B \in \mathbf{Blocks}$, $M, M', N \in \mathbf{Exp}_\mathcal{V}$:
(i) No pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$; that is, keys, blocks, pairs, ciphers are distinguishable.
(ii) If $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$.
Let M and N be valid formal expressions. Let $\mathcal{C}_M = \{C_M^1, ...C_M^{c(M)}\}$ be an enumeration of all ciphers encrypted by recoverable keys in M such that they can be decrypted in this order. Then, $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ implies that $c(M) = c(N)$, and $\mathcal{C}_N = \{C_N^1, ..., C_N^{c(N)}\}$ can be enumerated in the order of decryption such that $\Gamma_{c(M)}(N, M)C_M^i = C_N^i$. Moreover, with this enumeration of $\mathcal{C}_N$, $\mathcal{D}_i(M) = \mathcal{D}_i(N)$, and

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi)$$

Proof. Let M and N be expressions such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. Since we assumed condition (i) and since the equivalence $\approx$ is assumed to be invariant under depairing, the pairs that are not encrypted in M and in N must be in the same positions, and so $\mathcal{B}(M) = \mathcal{B}(N)$ must hold. Since the blow-up function is received by repeated application of the inverse of the pairing function, projecting and coupling,

$$\mathcal{B}(M)([\![M]\!]_\Phi) \approx \mathcal{B}(N)([\![N]\!]_\Phi). \tag{3.1}$$

As mentioned in Proposition 3.5, if x is sampled from $[\![M]\!]_\Phi$, then $\mathcal{B}(M)x \in \mathcal{T}_0(M)$. Therefore,

$$\mathcal{T}_0(M) = \mathcal{T}_0(N).$$

Since $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, there is a unique bijection

$$\Gamma_0(N, M) : sub_0(M) \rightarrow sub_0(N)$$

that satisfies

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(\Gamma_0(N, M)M', N).$$

Let $C^1_M = \{C^1_{M,\text{text}}\}_{C^1_{M,\text{key}}}$. Consider $L_1 := \Gamma_0(N, M)C^1_{M,\text{key}}$. L_1 must be a key for the following reason:

$$(\mathcal{G}_0(C^1_{M,\text{key}}, M) \circ \mathcal{B}(M))([\![M]\!]_\Phi) \approx (\mathcal{G}_0(C^1_{M,\text{key}}, M) \circ \mathcal{B}(M))([\![N]\!]_\Phi),$$

since we again apply the same function, $\mathcal{G}_0(C^1_{M,\text{key}}, M) \circ \mathcal{B}(M)$ on $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$, and this function is made up of depairing, projecting and coupling. But, for the left hand side we clearly have

$$(\mathcal{G}_0(C^1_{M,\text{key}}, M) \circ \mathcal{B}(M))([\![M]\!]_\Phi) = [\![C^1_{M,\text{key}}]\!]_\Phi,$$

and for the right hand side,

$$(\mathcal{G}_0(C^1_{M,\text{key}}, M) \circ \mathcal{B}(M))([\![N]\!]_\Phi) = (\mathcal{G}_0(L_1, N) \circ \mathcal{B}(N))([\![N]\!]_\Phi) = [\![L_1]\!]_\Phi.$$

Therefore, by assumption (i) L_1 must be a key. Similarly,

$$(\mathcal{G}_0(C^1_M, M) \circ \mathcal{B}(M))([\![M]\!]_\Phi) \approx (\mathcal{G}_0(C^1_M, M) \circ \mathcal{B}(M))([\![N]\!]_\Phi).$$

The left-hand side equals $[\![C^1_M]\!]_\Phi$, hence we need to have an interpretation of a cipher on the right too, implying that for some N' expression and L key,

$$\Gamma_0(N, M)C^1_M = \{N'\}_L$$

and hence

$$\mathcal{G}_0(C^1_M, M) = \mathcal{G}_0(\{N'\}_L, N). \tag{3.2}$$

Then, according to the foregoing,

$$\big(\mathcal{G}_0(C^1_{M,\mathrm{key}}, M), \mathcal{G}_0(C^1_M, M)\big) \circ \mathcal{B}(M)) = \big(\mathcal{G}_0(L_1, N), \mathcal{G}_0(\{N'\}_L, N)\big) \circ \mathcal{B}(N),$$

and therefore,

$$\Big(\big(\mathcal{G}_0(C^1_{M,\mathrm{key}}, M), \mathcal{G}_0(C^1_M, M)\big) \circ \mathcal{B}(M)\Big)\big([\![M]\!]_\Phi\big) \approx \Big(\big(\mathcal{G}_0(L_1, N), \mathcal{G}_0(\{N'\}_L, N)\big) \circ \mathcal{B}(N)\Big)\big([\![N]\!]_\Phi\big).$$

But, the left-hand side equals $[\![(C^1_{M,\mathrm{key}}, C^1)]\!]_\Phi$, whereas the right-hand side is $[\![(L_1, \{N'\}_L)]\!]_\Phi$, so we have

$$[\![(C^1_{M,\mathrm{key}}, C^1_M)]\!]_\Phi \approx [\![(L_1, \{N'\}_L)]\!]_\Phi.$$

By assumption (ii) then, $L = L_1$ follows, because $C^1_M = \{C^1_{M,\mathrm{text}}\}_{C^1_{M,\mathrm{key}}}$. But then we can choose the first element of $\mathcal{C}_N$ to be the occurrence $\{N'\}_{L_1}$, and with this choice,

$$\mathcal{D}_1(M) = \mathcal{D}_1(N).$$

Therefore

$$\mathcal{D}_1(M)(\mathcal{B}(M)([\![M]\!]_\Phi)) \approx \mathcal{D}_1(N)(\mathcal{B}(N)([\![N]\!]_\Phi)),$$

and therefore,

$$\mathcal{T}_1(M) = \mathcal{T}_1(N),$$

because $\mathcal{D}_1(M)(\mathcal{B}(M)([\![M]\!]_\Phi))$ gives a distribution on $\mathcal{T}_1(M)$, and $\mathcal{D}_1(N)(\mathcal{B}(N)([\![N]\!]_\Phi))$ gives a distribution on $\mathcal{T}_1(N)$.

An argument similar to the one above shows that

$$\mathcal{D}_2(M) = \mathcal{D}_2(N).$$

Namely, there is a unique bijection

$$\Gamma_1(N, M) : sub_1(M) \to sub_1(N)$$

satisfying

$$\mathcal{G}_1(M', M) = \mathcal{G}_1(\Gamma_1(N, M)M', N).$$

Then, just as we proved for L_1, $L_2 := \Gamma_1(N, M)C^2_{\text{key}}$ must be a key, and

$$\Gamma_1(N.M)C^2 = \{N''\}_{L_2}$$

for some N'' expression, implying that

$$\mathcal{D}_2(M) = \mathcal{D}_2(N).$$

And so on. So

$$\mathcal{D}_{c(M)}(M) \circ ... \circ \mathcal{D}_1(M)(\mathcal{B}(M)([\![M]\!]_\Phi)) \approx \mathcal{D}_{c(M)}(N) \circ ... \circ \mathcal{D}_1(N)(\mathcal{B}(N)([\![N]\!]_\Phi)),$$

since the functions applied on $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$ are the same, and they are made up only of depairing, projecting, coupling and decrypting. Then, $c(M) \leq c(N)$. Reversing the role of M and N in the argument, we get that $c(N) \leq c(M)$, and so $c(M) = c(N)$. Hence,

$$\mathcal{D}(M) = \mathcal{D}(N),$$

and

$$\mathcal{D}(M)([\![M]\!]_\Phi) = \mathcal{D}(N)([\![N]\!]_\Phi).$$

□

We would like to indicate how the proof goes via an example as well.

Example 3.21. Suppose again, that

$$M = \left(\left(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\right), \left(\left(K_2, \left\{(\{001\}_{K_3}, \{K_6\}_{K_5})\right\}_{K_5}\right), \{K_5\}_{K_2}\right)\right),$$

and assume that conditions (i) and (ii) of the lemma are satisfied. Suppose that N is also a valid

expression such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. Let

$$\begin{aligned} C_M^1 &= \{K_5\}_{K_2} \\ C_M^2 &= \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5} \\ C_M^3 &= \{K_6\}_{K_5} \\ C_M^4 &= \{0\}_{K_6}. \end{aligned}$$

M is a pair of two expressions: $M = (M_1, M_2)$. Then, since $[\![(M_1, M_2)]\!]_\Phi = [\![N]\!]_\Phi$, condition (i) of the lemma ensures that N must be a pair too: $N = (N_1, N_2)$. Then, since

$$[\![M_1]\!]_\Phi = \pi^1_{\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}} \circ [\cdot, \cdot]^{-1}([\![M]\!]_\Phi),$$

and

$$[\![N_1]\!]_\Phi = \pi^1_{\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}} \circ [\cdot, \cdot]^{-1}([\![N]\!]_\Phi)$$

(where $\pi^1_{\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}}$ denotes projection onto the first component of $\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}$), and since $\approx$ is assumed to be preserved by depairing and projecting, it follows that

$$[\![M_1]\!]_\Phi \approx [\![N_1]\!]_\Phi.$$

Therefore, since M_1 is a pair, N_1 must be a pair too. And so on. This way we conclude, that the unencrypted pairs in M are in the same position as the unencrypted pairs in N, and hence

$$\mathcal{B}(M) = \mathcal{B}(N).$$

It also follows then, that

$$\mathcal{T}_0(M) = \Big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\Big) \times \Big(\big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\big) \times \overline{\mathbf{strings}}\Big) = \mathcal{T}_0(N).$$

At this point, we know that N has the form

$$N = \Big(\big(N_3, N_4\big), \big((N_5, N_6), N_7\big)\Big)$$

Now, we took C^1_M to be $\{K_5\}_{K_2}$, the corresponding string, which is a cipher, is located in the last component of $\mathcal{T}_0(M)$. The key string that decrypts this cipher is located in the third component of $\mathcal{T}_0(M)$. Hence

$$\mathcal{G}_0(C^1_M, M) = \pi^5_{\mathcal{T}_0(M)}$$

and

$$\mathcal{G}_0(C^1_{M,\text{key}}, M) = \pi^3_{\mathcal{T}_0(M)}.$$

But then, since $\pi^i_{\mathcal{T}_0(M)}$ preserves $\approx$, it follows that

$$\pi^3_{\mathcal{T}_0(M)}(\mathcal{B}(M)([\![M]\!]_\Phi)) \approx \pi^3_{\mathcal{T}_0(M)}(\mathcal{B}(N)([\![N]\!]_\Phi)).$$

It is also true that

$$\pi^3_{\mathcal{T}_0(N)} = \mathcal{G}_0(N_5, N).$$

But

$$\mathcal{G}_0(C^1_{M,\text{key}}, M)(\mathcal{B}(M)([\![M]\!]_\Phi)) = [\![K_2]\!]_\Phi,$$

and

$$\mathcal{G}_0(N_5, N)(\mathcal{B}(N)([\![N]\!]_\Phi)) = [\![N_5]\!]_\Phi,$$

so

$$[\![N_5]\!]_\Phi \approx [\![K_2]\!]_\Phi,$$

and hence, by the assumption (i) of the lemma, it follows that N_5 must also be a key, let us denote it with L_1. Similarly,

$$\pi^5_{\mathcal{T}_0(N)} = \mathcal{G}_0(N_7, N),$$

but then

$$[\![N_7]\!]_\Phi \approx [\![\{K_5\}_{K_2}]\!]_\Phi,$$

and therefore N_7 must be a cipher: $N_7 = \{N'\}_L$ for some expression N' and key L. To get that $L = L_1$, consider

$$(\pi^3_{\mathcal{T}_0(M)}, \pi^5_{\mathcal{T}_0(M)}) \circ \mathcal{B}(M)([\![M]\!]_\Phi) = [\![(K_2, \{K_5\}_{K_2})]\!]_\Phi$$

and

$$(\pi^3_{\mathcal{T}_0(N)}, \pi^5_{\mathcal{T}_0(N)}) \circ \mathcal{B}(N)([\![N]\!]_\Phi) = [\![(L_1, \{N'\}_L)]\!]_\Phi.$$

From this, since the left-hand sides are equivalent, we conclude that

$$[\![(K_2, \{K_5\}_{K_2})]\!]_\Phi \approx [\![(L_1, \{N'\}_L)]\!]_\Phi,$$

which means by condition (ii) of the lemma that

$$L = L_1.$$

Therefore, if we define C^1_N as $\{N'\}_L$, then they and the keys that decrypt them are also in the same position, so

$$\mathcal{D}_1(M) = \mathcal{D}_1(N).$$

Remember from example 3.12, that $\mathcal{D}_1(M) = \mathcal{D}_1(N)$ does the following:

$$\mathcal{D}_1(M)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = \begin{cases} \Big(\big(x_1, x_2\big), \Big((x_3, x_4), \big(x_3, 0, \mathcal{B}(\{K_5\}_{K_2})(D_{x_3}(x_5))\big)\Big)\Big) \text{ if} \\ \qquad (x_3, x_5) \in \mathrm{Dom}_D \\ \big((x_1, x_2), ((x_3, x_4), (0, 0, 0)\big) \quad \text{otherwise,} \end{cases}$$

so if x is sampled from $[\![M]\!]_\Phi$ or $[\![N]\!]_\Phi$, then $\mathcal{D}_1(M)(\mathcal{B}(M)x) = \mathcal{D}_1(N)(\mathcal{B}(N)x)$ has the form

$$\Big(\big(x_1, x_2\big), \Big((x_3, x_4), \big(x_3, 0, x_6\big)\Big)\Big),$$

and

$$\mathcal{T}_1(M) = \mathcal{T}_1(N) = \Big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\Big) \times \Big(\big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\big) \times \big(\overline{\mathbf{strings}} \times \{0\} \times \overline{\mathbf{strings}}\big)\Big)$$

Then, continue this process until you show that $\mathcal{D}_4(M) = \mathcal{D}_4(N)$. □

3.2 Completeness

In this section, we prove some completeness results with the help of lemma 3.20. Just as in the case of soundness, we start with completeness for the type-2 case and the One-Time Pad in section 3.2.1 and section 3.2.2 respectively. Then, in section 3.2.3 we prove a general completeness theorem.

3.2.1 Completeness of Type-2 Encryption Schemes

We remind the reader, that a type-2 encryption scheme has the characteristic that

$$\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : \mathcal{A}_\eta^{\mathcal{E}_k(\mathbf{0})} = 1\right]$$

is a negligible function of η. This does not require that key-repetition is concealed. On the other hand, it may be concealed, the definition does not forbid that. We proved the soundness result for the logic expanded with boxes indexed by keys. By introducing these boxes, we ensured that formal ciphers were equivalent only if they were encrypted with the same key, and that was enough to prove computational indistinguishability after interpretation. Now we want to show that computational equivalence implies the formal, that is, formal *inequivalence* implies computational *inequivalence*. But since in our logic, ciphers with different encrypting keys are inequivalent, we have to make it sure that they have inequivalent interpretations. For completeness, we will therefore need the following property of the encryption scheme:

Definition 3.22 (Strictly Which-Key Revealing Scheme)**.** We say that an encryption scheme Π is *strictly which-key revealing* if it is type-2 secure and there exists a polynomial-time adversary A such that the following function is not negligible as a function of η:

$$\mathrm{Adv}_{\Pi[\eta]}(A) := \Pr\left[k, k' \leftarrow \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot),\mathcal{E}_{k'}(\cdot)} = 1\right] - \Pr\left[k \leftarrow \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot),\mathcal{E}_k(\cdot)} = 1\right]$$

□

Proposition 3.23. Suppose the cryptosystem is strictly which-key revealing. Then, for any

expressions M_1, M_2, N_1, N_2 and keys L_1, L_2, L

$$[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_L, \{N_2\}_L)]\!]_\Phi$$

implies

$$L_1 = L_2.$$

Proof. Assume the contrary. Since we assumed that the system is which-key revealing, there is an adversary A such that

$$\Pr\left[k, k' \leftarrow \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)} = 1\right] - \Pr\left[k \leftarrow \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_k(\cdot)} = 1\right]$$

is not negligible. But, by adding and subtracting a few terms, we can arrive at

$$\begin{aligned}
\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)} = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_k(\cdot)} = 1\right] = \\
\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)} = 1\right] - \\
-\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta,\ x_1 \xleftarrow{R} [\![M_1]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(x_1), \mathcal{E}_{k'}(\cdot)} = 1\right] + \\
+\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta,\ x_1 \xleftarrow{R} [\![M_1]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(x_1), \mathcal{E}_{k'}(\cdot)} = 1\right] - \\
-\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta,\ x_1 \xleftarrow{R} [\![M_1]\!]_{\Phi_\eta},\ x_2 \xleftarrow{R} [\![M_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(x_1), \mathcal{E}_{k'}(x_2)} = 1\right] + \\
+\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta,\ x_1 \xleftarrow{R} [\![M_1]\!]_{\Phi_\eta},\ x_2 \xleftarrow{R} [\![M_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(x_1), \mathcal{E}_{k'}(x_2)} = 1\right] - \\
-\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta,\ y_1 \xleftarrow{R} [\![N_1]\!]_{\Phi_\eta},\ y_2 \xleftarrow{R} [\![N_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(y_1), \mathcal{E}_k(y_2)} = 1\right] + \\
+\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta,\ y_1 \xleftarrow{R} [\![N_1]\!]_{\Phi_\eta},\ y_2 \xleftarrow{R} [\![N_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(y_1), \mathcal{E}_k(y_2)} = 1\right] - \\
-\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0})} = 1\right] + \\
+\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0})} = 1\right] - \\
-\Pr\left[(k \xleftarrow{R} \mathcal{K}_\eta : A_\eta^{\mathcal{E}_k(\cdot), \mathcal{E}_k(\cdot)} = 1\right]
\end{aligned}$$

Since the left hand side of this equation is supposed to be non-negligible, the right-hand side must also be non-negligible. But, on the right-hand side, the first, second, fourth and fifth differences

are negligible, because of type-2 security, therefore, the third difference, namely,

$$\Pr\left[k, k' \xleftarrow{R} \mathcal{K}_\eta,\ x_1 \xleftarrow{R} [\![M_1]\!]_{\Phi_\eta},\ x_2 \xleftarrow{R} [\![M_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(x_1),\mathcal{E}_{k'}(x_2)} = 1\right] -$$
$$-\Pr\left[k \xleftarrow{R} \mathcal{K}_\eta,\ y_1 \xleftarrow{R} [\![N_1]\!]_{\Phi_\eta},\ y_2 \xleftarrow{R} [\![N_2]\!]_{\Phi_\eta} : A_\eta^{\mathcal{E}_k(y_1),\mathcal{E}_k(y_2)} = 1\right]$$

must not be negligible, which proves the lemma. □

The following theorem characterizes the necessary requirements for a type-2 encryption scheme for the formal language that we introduced for the type-2 case and its interpretation to be complete. Condition (ii) in the theorem was used by Horvitz and Gligor in [26] when they were proving completeness for the type-0 case and they called it *weak confusion freeness*. Observe, that if the cryptosystem is strictly which-key revealing, then condition (iii) is satisfied according to the previous proposition.

Theorem 3.24. Let $\Pi = (\mathcal{K}, E, D, \approx)$ be a type-2 encryption scheme, Φ the interpretation in Π of $\Delta = (\mathbf{Exp}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$, where $\equiv_{\mathbf{K}}$ makes all keys equivalent, and $\equiv_{\mathbf{C}}$ means equivalence iff the encrypting keys are identical. Then, completeness of Φ holds if and only if
for any $K, K', K'' \in \mathbf{Keys}$, $B \in \mathbf{Blocks}$, $M, M', N \in \mathbf{Exp}_{\mathcal{V}}$,
(i) no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$; that is, keys, blocks, pairs, ciphers are distinguishable,
(ii) if $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$, and
(iii) for any expressions M_1, M_2, N_1, N_2 and keys L_1, L_2, L

$$[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_L, \{N_2\}_L)]\!]_\Phi$$

implies

$$L_1 = L_2.$$

Proof. The only if direction is trivial. In order to prove the if part, consider two expressions M and N such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. By condition (i) and (ii), Lemma 3.20 is applicable, so, $c(M) = c(N)$,

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi),$$

and

$$\mathcal{T}_{c(M)}(M) = \mathcal{T}_{c(N)}(N).$$

In each entry of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, the distribution corresponds either to the interpretation of a key, or of a block, or of an undecryptable cipher (*i.e.* one that corresponds to a box). Naturally, the same blocks must be in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, because the distributions $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, and because of condition (i). Hence, the patterns of M and N contain the same blocks in the same positions. Moreover, since $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, the entries in $\mathcal{T}_{c(M)}(M)$ and in $\mathcal{T}_{c(N)}(N)$ containing strings sampled from key generation must be in the same places because of (i) again. Furthermore, the indistinguishability of in $\mathcal{T}_{c(M)}(M)$ and in $\mathcal{T}_{c(N)}(N)$ also implies that repetitions of a key generation outcome must occur in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$ as well. (This is a consequence of the properties of key-generation in definition 1.36.) Therefore the key symbols in the patterns of M and N change together, so it is possible to rename the recoverable keys of N so that the keys in the pattern of $N\sigma$ are the same as the keys in the pattern of M. Finally, since the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, condition (i) implies that pairs of indistinguishable undecryptable ciphers occur in exactly the same entries in $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$. But, that means, according to condition (iii), that if there are undecryptable ciphers encrypted with identical (or different) keys in M, then in the same position in N, there are undecryptable ciphers encrypted with identical (or different) keys. Hence, the identical boxes in the pattern of M, located in exactly the same places as the identical boxes in the pattern of N and hence of $N\sigma$ (because renaming the recoverable keys does not effect the boxes; they are indexed with non-recoverable keys). But then it is possible to rename those keys that appear as box-indexes (keys in *B-Keys*$(N\sigma)$), so that the box-indexes in the two patterns agree as well. Therefore, the two patterns must agree up to key-renaming. □

In order to see it more clearly how the proof works, we introduce an example:

Example 3.25. Consider, as we did earlier in this chapter, the expression

$$M = \Big(\big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\big), \Big(\big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big),$$

with pattern

$$pattern_2(M) = \Big(\big(\{0\}_{K_6}, \Box_{K_4}\big), \Big(\big(K_2, \big\{(\Box_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\big), \{K_5\}_{K_2}\Big)\Big),$$

and assume, that N is another expression such that

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

We have to show, that $N \cong_2 M$. According to lemma 3.20, $\mathcal{D}(M) = \mathcal{D}(N)$, and, as we saw it in the examples of section 3.1, if y is a sample from $[\![M]\!]_\Phi$ or of $[\![N]\!]_\Phi$, then $\mathcal{D}(M)y = \mathcal{D}(N)y$ has the form

$$\Big(\big((y_6, 0, 0), y_1\big), \Big(\big(y_2, \big(y_5, 0, (y_3, (y_5, 0, y_6))\big)\big), \big(y_2, 0, y_5\big)\Big)\Big), \tag{3.3}$$

where, if y is sampled from $[\![M]\!]_\Phi$, then y_2, y_5, y_6 are outcomes of the key-generation algorithms $\phi(K_2)$, $\phi(K_5)$, $\phi(K_6)$ respectively, y_1 is an undecryptable sample element from $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$, and y_3 is an undecryptable sample from $[\![\{001\}_{K_3}]\!]_\Phi$.

For the case when y is sampled from $[\![N]\!]_\Phi$ the form of expression 3.3 shows that counting from the left, the first occurrence of y_6, the first and second occurrence of y_5, and the second occurrence of y_2 must be samples from interpretations of keys. The different occurrences of these elements vary together as we sample other elements, so, since we assumed that the distribution of separate key generations can be distinguished from the same key generation twice (in definition 1.36), we get that the different occurrences of y_2, y_5 and y_6 must stand for the same subexpression keys of N. Let us call them L_1, L_2 and L_3 respectively. L_1, L_2 and L_3 must be all different keys, because y_2, y_5 and y_6 vary separately. Formula 3.3 shows then that N has the following form:

$$N = \Big(\big(\{N_1\}_{L_3}, N_2\big), \Big(\big(L_1, \big\{(N_3, \{L_3\}_{L_2})\big\}_{L_2}\big), \{L_2\}_{L_1}\Big)\Big).$$

Comparing this with 3.3, we also have that 0 is a sample element of $[\![N_1]\!]_\Phi$, y_1 is a sample element from $[\![N_2]\!]_\Phi$, and y_3 is a sample from $[\![N_3]\!]_\Phi$. Therefore, since

$$\mathcal{D}(M)([\![M]\!]_\Phi) = \mathcal{D}(N)([\![N]\!]_\Phi), \tag{3.4}$$

it follows that

$$[\![N_1]\!]_\Phi \approx [\![0]\!]_\Phi,$$

$$[\![N_2]\!]_\Phi \approx [\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi,$$

$$[\![N_3]\!]_\Phi \approx [\![\{001\}_{K_3}]\!]_\Phi.$$

We assumed that the conditions of the theorem holds, so (i) implies that for some N_4 and N_5 subexpressions and L_4, L_5 keys,

$$N_1 = 0,$$

$$N_2 = \{N_4\}_{L_4},$$

$$N_3 = \{N_5\}_{L_5}.$$

L_4 and L_5 cannot equal any of L_1, L_2 or L_3, because otherwise we could have continued the decryption process for N, and therefore for M as well. Equation 3.4 also implies that

$$[\![(\{\{K_7\}_{K_1}\}_{K_4}, \{001\}_{K_3}]\!]_\Phi \approx [\![(\{N_4\}_{L_4}, \{N_5\}_{L_5})]\!]_\Phi,$$

and therefore, by condition (iii), we receive $L_4 \neq L_5$. Hence, up to key-renaming, N looks like

$$N' = \Big(\big(\{0\}_{K_3}, \{N_4\}_{K_4}\big), \Big(\big(K_1, \big\{(\{N_5\}_{K_5}, \{K_3\}_{K_2})\big\}_{K_2}\big), \{K_2\}_{K_1}\Big)\Big),$$

which has pattern

$$\mathit{pattern}(N') = \Big(\big(\{0\}_{K_3}, \square_{K_4}\big), \Big(\big(K_1, \big\{(\square_{K_5}, \{K_3\}_{K_2})\big\}_{K_2}\big), \{K_2\}_{K_1}\Big)\Big),$$

which shows that

$$M \cong_2 N.$$

□

3.2.2 Completeness for One-Time Pad

In this section we show completeness for the One-Time Pad. The method of the proof is quite similar to the proof method in the type-2 case. Observe though, that conditions like (i)-(iii) in theorem 3.24 here are missing. The reason is, that we considered a specific implementation of the One-Time Pad, whereas the fact that an encryption scheme is type-2, does not specify the scheme completely; our specific OTP implementation does satisfy the corresponding conditions, as we will see it in the course of the proof.

Theorem 3.26. Let $\Pi = (\{\mathcal{K}_n\}_{n\geq 4}, E, D, \approx)$ be the One-Time Pad encryption scheme that we presented in Section 1.3.1, Φ the interpretation of $\mathbf{Exp}_{\mathrm{OTP}}$ for the OTP. Then, completeness of Φ holds.

Proof. Consider two expressions M and N such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. In order to be able to apply Lemma 3.20, we need to show that conditions (i) and (ii) of the lemma hold for One-Time Pad. The reason for this is that the samples of the interpretations of expressions are tagged, that is, $[\![M]\!]_\Phi$ has a tag, which is either 001, 010, 001, or ends with 1 in case of pairs. So, keys, blocks, pairs, ciphers are clearly distinguishable, and hence condition (i) holds. To see that condition (ii) is satisfied, consider $Q = (K, \{M\}_K)$ and $Q' = (K'', \{M'\}_{K'})$. Since the last digit of

$$D_{\Phi_Q(K)(\omega)}(\Phi_Q(\{M\}_K)(\omega))$$

is constant for all ω, whereas the last digit of

$$D_{\Phi_{Q'}(K'')(\omega)}(\Phi_{Q'}(\{M'\}_{K'})(\omega))$$

changes (probability 0.5 for 1 and 0.5 for 0) if $K' \neq K''$, condition (ii) is satisfied. Hence, we can apply the lemma.

By Lemma 3.20 then, $c(M) = c(N)$,

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi),$$

and

$$\mathcal{T}_{c(M)}(M) = \mathcal{T}_{c(N)}(N).$$

In each entry of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, the distribution corresponds either to the interpretation of a key, or of a block, or of an undecryptable cipher (*i.e.* one that corresponds to a box). Naturally, the same blocks must be in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, because the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are identical . Hence, the patterns of M and N contain the same blocks in the same positions. Moreover, the entries with identical keys (those which are the same outcomes of a key-generation) in $\mathcal{T}_{c(M)}(M)$ must occur exactly in the same places as the identical keys occur $\mathcal{T}_{c(N)}(N)$, therefore the key symbols in the patterns of M and N change together, so it is possible to rename the recoverable keys of N with length-preserving σ so that the keys in the pattern of $N\sigma$ are the same as the keys in the pattern of M. Finally, since the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are identical, ciphers with the same length occur in the same entries of $\mathcal{T}_{c(M)}(M)$ as in $\mathcal{T}_{c(N)}(N)$. Therefore, boxes of the same length will appear in the same position in the two patterns, and that is what we needed to prove. □

3.2.3 General Case

Theorem 3.27. Let $\Delta = (\mathbf{Exp}_\mathcal{V}, \equiv_\mathbf{K}, \equiv_\mathbf{C})$ be a formal logic for symmetric encryption, assume that $\equiv_\mathbf{C}$ is proper and that $\equiv_\mathbf{K}$ and $\equiv_\mathbf{C}$ are independent. Let Φ be an interpretation in $\Pi = (\{\mathcal{K}_i\}_{i\in I}, E, D, \approx)$. Then, completeness of Φ holds, if and only if the following conditions are satisfied : For any $K, K', K'' \in \mathbf{Keys}$, $B \in \mathbf{Blocks}$, $M, M', N \in \mathbf{Exp}_\mathcal{V}$,

(i) no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$; that is, keys, blocks, pairs, ciphers are distinguishable.

(ii) if $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$.

(iii) For any two pairs of valid ciphers, $\{\{M_i\}_{L_i}\}_{i=1}^2$, $\{\{N_i\}_{L'_i}\}_{i=1}^2$

$$[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_{L'_1}, \{N_2\}_{L'_2})]\!]_\Phi$$

implies

$$(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong_\Delta (\{N_1\}_{L'_1}, \{N_2\}_{L'_2}).$$

Proof. The only if part is trivial. In order to prove the if part, consider two expressions M and N such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. By condition (i) and (ii), Lemma 3.20 is applicable, so, $c(M) = c(N)$,

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi),$$

and

$$\mathcal{T}_{c(M)}(M) = \mathcal{T}_{c(N)}(N).$$

In each entry of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, the distribution corresponds either to the interpretation of a key, or of a block, or of an undecryptable cipher (*i.e.* one that corresponds to a box). Naturally, the same blocks must be in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, because the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, and because of condition (i). Hence, the patterns of M and N contain the same blocks in the same positionsMoreover, since $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, the entries in $\mathcal{T}_{c(M)}(M)$ and in $\mathcal{T}_{c(N)}(N)$ containing strings sampled from key generation must be in the same places because of (i) again. Furthermore, the indistinguishability of in $\mathcal{T}_{c(M)}(M)$ and in $\mathcal{T}_{c(N)}(N)$ also implies that repetitions of a key generation outcome must occur in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$ as well. (This is a consequence of the properties of key-generation in definition 1.41.) Therefore the key symbols in the patterns of M and N change together, so it is possible to rename the recoverable keys of N (with a $\equiv_{\mathbf{K}}$ preserving function σ so that the keys in the pattern of $N\sigma$ are the same as the keys in the pattern of M.

Since the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, condition (i) implies that the undecryptable ciphers occur in exactly the same entries in $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$. This means, that in the pattern of M and N, the boxes appear in the same position. This together with the conclusions of the previous paragraph means, that apart from the boxes, everything else in the pattern of M and of $N\sigma$ must be the same. By replacing N with $N\sigma$, we can assume from now on that the recoverable keys of N and M are identical, and that the pattern of M and N are the same outside the boxes. Therefore, we only have to show that there is a key renaming σ' that carries the boxes of N into the boxes of M without changing the recoverable keys.

Say, there are l boxes altogether in the pattern of M (and hence in the pattern of N). Let

$\{M_1\}_{L_1}$, $\{M_2\}_{L_2}$, ..., $\{M_l\}_{L_l}$ be the corresponding undecryptable ciphers in M that turn into boxes in M and $\{N_1\}_{L'_1}$, $\{N_2\}_{L'_2}$, ..., $\{N_l\}_{L'_l}$ the corresponding ciphers in N. Then, for $i, j \leq l$, $i \neq j$,

$$[\![(\{M_i\}_{L_i}, \{M_j\}_{L_j})]\!]_\Phi \approx [\![(\{N_i\}_{L'_i}, \{N_j\}_{L'_j})]\!]_\Phi$$

holds since $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable. This means by condition (iii), that

$$(\{M_i\}_{L_i}, \{M_j\}_{L_j}) \cong_\Delta (\{N_i\}_{L'_i}, \{N_j\}_{L'_j})$$

and hence there is a key-renaming σ_{ij} such that

$$\sigma_{ij}(\mu(\{N_i\}_{L'_i})) = \mu(\{M_i\}_{L_i})$$

and

$$\sigma_{ij}(\mu(\{N_j\}_{L'_j})) = \mu(\{M_j\}_{L_j}),$$

implying

$$(\square_{\mu(\{M_i\}_{L_i})}, \square_{\mu(\{M_j\}_{L_j})}) = (\square_{\sigma_{ij}(\mu(\{N_i\}_{L'_i}))}, \square_{\sigma_{ij}(\mu(\{N_j\}_{L'_j}))}).$$

We assumed that $\equiv_{\mathbf{C}}$ is proper, therefore, by proposition 1.32, each μ box-index of N has a representative C_μ such that C_μ does not contain any element of $R\text{-}Keys(N) = R\text{-}Keys(M)$, and for any two different μ_1 and μ_2, the only common element of $Keys(C_{\mu_1})$ and $Keys(C_{\mu_2})$ may be the encrypting key, only if there is one possible encrypting key for all elements in μ_1 and μ_2. We define σ' inductively. Since we assumed that $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$ are independent, it is possible to modify σ_{12} such that the σ_1 that we get leaves

$$\left(\bigcup_{i=3}^{l} Keys(C_{\mu(\{N_i\}_{L'_i})}) \cup R\text{-}Keys(N)\right) \setminus \left(\{L'_1, L'_2\} \cup Keys(C_{\mu(\{N_1\}_{L'_1})}) \cup Keys(C_{\mu(\{N_2\}_{L'_2})})\right)$$

untouched but still

$$\sigma_1(\mu(\{N_1\}_{L'_1})) = \mu(\{M_1\}_{L_1})$$

and

$$\sigma_1(\mu(\{N_2\}_{L'_2})) = \mu(\{M_2\}_{L_2})$$

hold. Suppose now that we have defined σ_k such that for some h, $h > k$,

$$\sigma_k(K) = K,$$

whenever

$$K \in \left(\bigcup_{i=h+1}^{l} Keys(C_{\mu(\{N_i\}_{L'_i})}) \cup R\text{-}Keys(N) \right) \setminus \left(\{L'_1, ..., L'_h\} \cup \left(\bigcup_{i=1}^{h} Keys(C_{\mu(\{N_i\}_{L'_i})}) \right) \right),$$

but

$$\sigma_k(\mu(\{N_i\}_{L'_i})) = \mu(\{M_i\}_{L_i})$$

for $i \leq h$. In order to define σ_{k+1}, find first the smallest $a > h$ such

$$C_{\mu(\{N_a\}_{L'_a})} \neq C_{\mu(\{N_i\}_{L'_i})}$$

holds for all $i < a$. If there is no such a, then the process stops, and $\sigma' := \sigma_k$. If there is such an a, then consider σ_{aj} with any $j \neq a$. Since $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$ are independent, it is possible to alter σ_{aj} into σ'_{aj} such that

$$\sigma'_{aj}(\mu(\{N_a\}_{L'_a})) = \mu(\{M_a\}_{L_a})$$

and

$$\sigma'_{aj}(K) = K$$

whenever

$$K \in \bigcup_{i=1}^{l} Keys(C_{\mu(\{N_i\}_{L'_i})}) \cup \sigma_k \left(\bigcup_{i=1}^{l} Keys(C_{\mu(\{N_i\}_{L'_i})}) \right) \cup R\text{-}Keys(N)$$

and

$$K \notin \{L'_a\} \cup Keys(C_{\mu(\{N_a\}_{L'_a})}).$$

Therefore, the only key K in $\bigcup_{i=1}^{l} Keys(C_{\mu(\{N_i\}_{L'_i})})$ that is changed by both σ_k and σ'_{aj} may only be L'_a, and that happens by propositions 1.30 and 1.32 when for some $i \leq h$,

$$\mu(\{N_a\}_{L'_a})_{\text{key}} = \mu(\{N_i\}_{L'_i})_{\text{key}} = \{L'_a\} = \{L'_i\}.$$

Since

$$\sigma'_{aj}(\mu(\{N_a\}_{L'_a})) = \mu(\{M_a\}_{L_a}),$$

it follows by proposition 1.31 that $\mu(\{M_a\}_{L_a})_{\text{key}} = \{L_a\}$, and

$$\sigma'_{aj}(L'_a) = L_a$$

must hold. Also, since

$$\sigma_{ia}(\mu(\{N_i\}_{L'_i})) = \mu(\{M_i\}_{L_i}),$$

$$\sigma_{ia}(L'_i) = L_i$$

follows, and

$$\sigma_{ia}(\mu(\{N_a\}_{L'_a})) = \mu(\{M_a\}_{L_a})$$

implies

$$\sigma_{ia}(L'_a) = L_a.$$

But, $L'_i = L'_a$, so

$$L_i = L_a.$$

Then,

$$\sigma_k(L'_a) = \sigma_k(L'_i) = \sigma_{ia}(L'_i) = L_i = L_a.$$

Hence, σ'_{ia} and σ_k take the same value on L'_a which is the only interesting element that they both change. So we can define σ_{k+1} to be the key-renaming that does the job of both σ'_{ia} and σ_k together. This way, for $a > k+1$,

$$\sigma_{k+1}(K) = K,$$

whenever

$$K \in \left(\bigcup_{i=a+1}^{l} \mathit{Keys}(C_{\mu(\{N_i\}_{L'_i})}) \cup \mathit{R\text{-}Keys}(N) \right) \setminus \left(\{L'_1, ..., L'_a\} \cup \left(\bigcup_{i=1}^{a} \mathit{Keys}(C_{\mu(\{N_i\}_{L'_i})}) \right) \right),$$

but

$$\sigma_{k+1}(\mu(\{N_i\}_{L'_i})) = \mu(\{M_i\}_{L_i})$$

for $i \leq a$. The process stops, when there is no $a \geq k$ such that

$$C_{\mu(\{N_a\}_{L'_a})} \neq C_{\mu(\{N_i\}_{L'_i})}$$

holds for all $i < a$, and then $\sigma' = \sigma_k$. This σ' satisfies the required properties, that is, it leaves the recoverable keys of M and N untouched, but it maps the boxes of the pattern of N into the corresponding boxes in the pattern of M, and that is what we needed to complete the proof. □

Remark 3.28. Observe, that condition (iii) of the theorem is is trivially satisfied, when there is only one box, that is, when all ciphers are equivalent under $\equiv_{\mathbf{C}}$. Also, if completeness holds for a certain choice of $\equiv_{\mathbf{C}}$, then, if $\equiv'_{\mathbf{C}}$ is such that $M \equiv_{\mathbf{C}} N$ implies $M \equiv'_{\mathbf{C}} N$ – i.e. when $\equiv'_{\mathbf{C}}$ results fewer boxes –, then completeness holds for $\equiv'_{\mathbf{C}}$ as well. Therefore, we can say, that the key to completeness is not to have too many boxes. □

Example 3.29 (Type-2 Cryptosystems)**.** Comparing this proof with the one for the type-two cryptosystems, it is clear that the theorem for type-2 systems is a special case of our general theorem. The formal language for the type-2 case is such that $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent as we mentioned in section 1.1.4. Moreover, conditions (i) and (ii) of the general theorem also appear in the conditions for type-2 completeness, whereas conditions (iii) in the two theorems are also identical for the type-2 case because of the following reason:

$$\mathit{pattern}_2((\{M_1\}_{L_1}, \{M_2\}_{L_2})) = (\square_{L_1}, \square_{L_2})$$

and

$$\mathit{pattern}_2((\{N_1\}_{L'_1}, \{N_2\}_{L'_2})) = (\square_{L'_1}, \square_{L'_2}),$$

hence $(\{M_1\}_{L_1}, \{M_2\}_{L_2})$ and $(\{M_1\}_{L_1}, \{M_2\}_{L_2})$ are formally equivalent if and only if L_1 and L_2 are identical exactly when L'_1 and L'_2 are identical. Therefore, conditions (iii) in the two theorems really mean the same thing. □

Example 3.30 (One-Time Pad)**.** The conditions of the general completeness theorem are satisfied by the formal language we use for the OTP, because, as we mentioned in section 1.1.4, $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent. Furthermore, condition (i) and (ii) in the general theorem are exactly conditions (i) and (ii) of the parsing theorem 3.20, and they are satisfied in the OTP case, as we saw that in the course of proving completeness for OTP. Condition (iii) is also satisfied, since the pairs of ciphers must be encrypted with different keys (in OTP, we cannot use the keys twice), and the equivalence

$$[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_{L_1'}, \{N_2\}_{L_2'})]\!]_\Phi$$

implies that the corresponding lengths in the two ciphers must be the same:

$$l(\{M_1\}_{L_1}) = l(\{N_1\}_{L_1'})$$

and

$$l(\{M_2\}_{L_2}) = l(\{N_2\}_{L_2'})$$

implying

$$\left(\Box_{l(\{M_1\}_{L_1})}, \Box_{l(\{M_2\}_{L_2})}\right) = \left(\Box_{l(\{N_1\}_{L_1'})}, \Box_{l(\{N_2\}_{L_2'})}\right).$$

Therefore,

$$(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong_{\text{OTP}} (\{N_1\}_{L_1'}, \{N_2\}_{L_2'}).$$

□

Example 3.31 (Type-1 and Type-3 Cryptosystems)**.** In case of Type-1 cryptosystems, if we assume that the length is revealed, that is the distributions of $E_k(x)$ and $E_k(y)$ can be distinguished when x and y have different length (we can call this condition strictly length revealing), then the corresponding condition (iii) is satisfied for this case. Therefore, if the cryptosystem is such that conditions (i) and (ii) are also satisfied, then completeness holds for the formal logic and its interpretation if the boxes are indexed with the length of the cipher.

As for the type-3 system, completeness holds if we assume that the system satisfies conditions (i) and (ii), and when it not just might reveal which-key and length, but it does really reveal

both them, that is, when it is strictly which-key revealing and strictly length revealing. □

Conclusion

We showed that it is possible to give a general treatment of expansions of the equivalence notion of the Abadi-Rogaway logic, and its interpretations in computational and also in information-theoretic encryption schemes. Non-trivial general soundness and completess theorems were established. These theorems derive soundness and completeness by assuming that they hold for special subsets of formal expressions. The key to soundness is to have enough boxes in the definition of formal equivalence, whereas completeness holds if there are not too many boxes.

The fact that the computational and the information-theoretic views give a more detailed description of cryptographic schemes then the formal one results in an interesting skewness of the conditions for soundness and for completeness. The conditions of completeness involves only pairs of formal keys, blocks and ciphers, reflecting that equivalence of formal expressions paired arbitrarily many times can be derived from equivalence of simple pairs. On the other hand, the conditions of soundness requires expressions that are constructed from keys, blocks, and ciphers via arbitrarily large number of pairing; the ultimate reason being that indistinguishability of the joint distributions of two n-tuples of random variables does not follow from indistinguishability of the joint distributions of each two corresponding pairs in the two n-tuples.

The Abadi-Rogaway logic, due to its simplicity, was very suitable to start analyzing the relationship between formal and probabilistic views of cryptography, but it is too simple for the description of realistic protocols. We therefore hope that this analysis will serve as a motivation and guideline for treating more complex formal cryptographic systems and their interpretations.

Bibliography

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi-calculus. *Information and Computation*, 143:1–70, 1999.

[2] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. Fourth International Symposium on Theoretical Aspects of Computer Software (TACS2001)*, Lecture Notes in Computer Science, Tohoku University, Sendai, Japan, 2001. Springer.

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

[4] M. Backes, C. Jacobi, and B. Pfitzmann. Deriving cryptographically sound implementations using composition and formally verified bisimulation. In *Formal Methods Europe*, volume 2931 of *Lecture Notes in Computer Science*, pages 310–329. Springer-Verlag, 2002.

[5] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security*, pages 220–230, Washington D.C., USA, October, 27-30 2003. ACM Press. Long version: IACR ePrint Archive, Report 2003/015, Jan. 2003.

[6] M. Backes, B. Pfitzmann, and M. Waidner. Universally composable cryptographic library. Manuscript available on eprint.iacr.org as 2003/015, 2003.

[7] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO '94,* 14^{th} *Annual International Cryptology Con-*

ference, volume 839 of *Lecture Notes in Computer Science*, pages 341–358, Santa Barbara, California, USA, August 1994. Springer-Verlag.

[8] M. Blum and S. Micali. Proceedings of the fourteenth annual ieee symposium on logic in computer science. In *Proc. of the 23rd Annual Symp. on Foundations of Computer Science*, pages 112–117, 1982.

[9] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society, Series A*, 426(1871):233–271, 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[10] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42-nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Press, 2001. Full paper available at eprint.iacr.org as 2000/067.

[11] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34-th ACM Symposium on Theory of Computing*, pages 484–503, 2002. Full paper available at eprint.iacr.org as 2002/140.

[12] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.

[13] R. A. DeMillo, N. A. Lynch, and M. Merritt. Cryptographic protocols. In *Proc. of the 14th Annual ACM Symp. on Theory of Computing*. ACM Press, 1982.

[14] D. Dolev and A. Yao. On the security of public-key protocols. In *Proc. 22-nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 350–357, 1981.

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

[16] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12:247–311, 2004.

[17] N. A. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, Canada, June 2001.

[18] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[19] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In Kwok Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *Advances in Cryptology — ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*, pages 232–246. Springer-Verlag, November 1999.

[20] O. Goldreich, S. Micali, and A. Widgerson. How to play any mental game. In *Proc. 19th ACM Symp. on the Theory of Computing*, pages 218–229, 1987.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. Previous version in *STOC 1982*.

[22] S. Goldwasser, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[23] J. Guttman, F. Thayer, and L. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In P. Samarati, editor, *Proceedings of the* 8^{th} *ACM conference on Computer and Communications Security*, pages 186–195, Philadelphia, Pennsylvania, USA, November, 5-8 2001. ACM Press.

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

[25] J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–567. Springer-Verlag, August 2003.

[26] O. Horvitz and V. Gligor. Weak key authenticity and the computational completeness of formal encryption. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003,* 23^{rd}

Annual International Cryptology Conference, volume 2729 of *Lecture Notes in Computer Science*, pages 530–547, Santa Barbara, California, USA, August, 17-21 2003. Springer-Verlag.

[27] R. A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Comunications*, 7(4):448–457, May 1989.

[28] R. A. Kemmerer, C. Meadows, and J. K. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[29] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In J. I. Lim and D. H. Lee, editors, *Information Security and Cryptology - ICISC 2003:* 6^{th} *International Conference*, volume 2971 of *Lecture Notes in Computer Science*, pages 55–66, Seoul, Korea, November, 27-28 2003. Springer-Verlag.

[30] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time framework for protocol analysis. In M. Reiter, editor, *5-th ACM Conferece on Computer and Communication Security*, pages 112–121. ACM Press, 1998.

[31] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In Tiziana Margaria and Bernhard Steffen, editors, *2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[32] P. Mateus, J. C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In Roberto M. Amadio and Denis Lugiez, editors, *14th International Conference on Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 327–349, Marseille, France, September 2003. Springer-Verlag.

[33] U. Maurer. Information-theoretic cryptography. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 47–64. Springer-Verlag, 1999.

[34] U. Maurer. Indistinguishability of random systems. In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer-Verlag, 2002. the extended version is not yet available.

[35] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *Theory of Cryptography — TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, February 2004.

[36] U. Maurer and S. Wolf. Information-theoretic key agreement: From weak to strong secrecy for free. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 351–368. Springer-Verlag, 2000.

[37] C. Meadows. A system for the specification and analysis of key management protocols. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 182–195. IEEE Computer Society Press, 1991.

[38] C. Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. In *Proc. European Symposium On Research In Computer Security*, pages 351–364. Springer Verlag, 1996.

[39] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004. Preliminary version presented at WITS'02.

[40] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Theory of Cryptography: First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, Massachusetts, USA, February, 19-21 2004. Springer-Verlag.

[41] J. K. Millen, S. C. Clark, and S. B. Freedman. The interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[42] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. *Electronic Notes in Theoretical Computer Science*, 45, 2001.

[43] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[44] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–9, 1978.

[45] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.

[46] L. C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.

[47] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. *Electronic Notes in Theoretical Computer Science*, 32, 2000.

[48] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7-th ACM Conference on Computer and Communications Security*, pages 245–254. ACM Press, 2000.

[49] A. W. Roscoe. Modeling and verifying key-exchange protocols using CSP and FDR. In *CSFW 8*, page 98. IEEE Computer Society Press, 1995.

[50] S. Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, Oakland, California, 1996.

[51] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[52] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[53] A. Yao. Theory and applications of trapdoor functions. In *23-rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE Press, 1982.