Computational and Information-Theoretic Soundness and Completeness of Formal Encryption

Pedro Adão *
Center for Logic and Computation
IST, Lisbon, Portugal
pad@math.ist.utl.pt

Gergei Bana †
Department of Mathematics
University of Pennsylvania, Philadelphia, USA
bana@math.upenn.edu

Andre Scedrov ‡
Department of Mathematics
University of Pennsylvania, Philadelphia, USA
scedrov@math.upenn.edu

Abstract

We consider expansions of the Abadi-Rogaway logic of indistinguishability of formal cryptographic expressions. We expand the logic in order to cover cases when partial information of the encrypted plaintext is revealed. 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 based on One-Time Pad, and computational settings of the so-called type-2 (which-key revealing) and type-3 (which-key and length revealing) encryption schemes based on computational complexity.

* Partially supported by FCT grant SFRH/BD/8148/2002. Additional support from FEDER/FCT project Fiblog POCTI/2001/MAT/37239 and FEDER/FCT project Quant-Log POCTI/MAT/55796/2004.

† Partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795. Additional support from NSF Grant CNS-0429689.

‡ Partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795 and OSD/ONR CIP/SW URI "Trustworthy Infrastructure, Mechanisms, and Experimentation for Diffuse Computing" through ONR Grant N00014-04-1-0725. Additional support from NSF Grants CCR-0098096 and CNS-0429689.

1. Introduction

Designing and verifying security protocols are complex problems; a 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 and hard to formulate. Checking whether protocols really satisfy the properties 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 approaches being a highly abstract treatment with the help of formal logic and a more detailed description using computational complexity and probability theory.

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

There have been several research efforts recently to relate the symbolic model of cryptographic techniques and the computational model based on probabilistic polynomial-time computability, including [15, 2, 5, 1, 18, 3, 11, 19, 9]. These efforts are developing rigorous mathematical treatment of the relationship between the two models.

The approach in [2], with which we are concerned here, uses a simple formal structure by building messages from formal keys and bits via repeated pairing and encrypting, constructing a set of formal expressions. These formal expressions are then interpreted in a computational framework of symmetric encryptions. Through this interpretation, an ensemble of probability distributions on the set of finite bit strings is assigned to each formal expression.

In each of the formal and the computational models, security is stated by means of a certain notion of equivalence. In the formal model, equivalence of symbolic expressions is defined inductively on the structure of expressions. In the computational model, equivalence of ensembles of probability distributions is given by the standard notion of computational indistinguishability [21]. The question is, what happens to the formal equivalence through the interpretation. If it is true that formal equivalence of any two symbolic expressions implies computational indistinguishability of their interpretations, then we say that soundness holds. If the other direction is true, namely, computational indistinguishability of the interpretations of any two symbolic expressions implies that the expressions are formally equivalent as well, we then say that completeness holds.

Related work includes the seminal work by Abadi and Rogaway[2], where they prove soundness in the case of so-called type-0 symmetric encryption schemes. Completeness, for the same case, was proved by Micciancio and Warinschi [18] and Horvitz and Gligor [11]. Extensions of the method include public-key encryption [19, 17], composite keys [14], plaintext-aware encryption schemes [9, 10], and signature schemes [6, 12].

Our Work. Our work extends applicability of the Abadi-Rogaway (AR) logic. By expanding the original AR logic, we show how to adjust the formal notion of equivalence in order to maintain soundness and completeness when the symmetric encryption scheme that hosts the interpretation (computational or information-theoretic) leaks partial information. That is, we show that distinctions among security levels of computational or information-theoretic encryption schemes can often be faithfully reflected in the symbolic model.

In order to provide a general treatment, we also consider interpretations in purely probabilistic, *information-theoretic* encryption schemes besides computational encryption schemes. We use a general probabilistic framework that includes as special cases both the computational and purely probabilistic encryption schemes (such as One-Time Pad). The advantage of this presentation is that there is no need to formulate general statements twice when they are true for both computational and information-theoretic models.

We prove general soundness and completeness theorems for our logics of formal symmetric encryptions. These theorems essentially claim that if soundness holds for a certain subset of the formal expressions, then soundness is valid for all expressions; similarly regarding completeness. As expected, it is necessary to assume soundness for a greater subset of expressions than for completeness in order to derive the theorems. The reason is that the probabilistic model is a more detailed description than the symbolic one: Indistinguishability of distributions of two n-tuples of random variables does not follow from indistinguishability of each two corresponding *pairs* in the n-tuples. In contrast, equivalence of two n-tuples of formal expressions can be derived from pairwise equivalence.

The rest of the paper is organized as follows. We start by presenting the Abadi and Rogaway formalism for logics of Formal Encryption. In Section 3 we introduce the fundamentals of the Computational Model. In Sections 4 and 5 we discuss Soundness and Completeness for type-2 encryption schemes and One-Time Pad. Finally, in Section 6 we present our general probabilistic framework, which includes both the computational and the information-theoretic encryption schemes as special cases. We prove our general soundness and completeness results and demonstrate the soundness and completeness of type-1, type-2, type-3 and OTP encryption schemes as corollaries of the general theorems. This is the main technical contribution of this paper. Finally, Section 7 concludes with a discussion of possible expansions of our logic as well as relations with other existing models.

We want to thank M. Abadi, J. Guttman, J. Herzog, R. Küsters, D. Micciancio, J. Mitchell and B. Warinschi for their valuable comments and informative discussions. This work was done while the first author was a visiting student at the University of Pennsylvania.

2. The AR Logic of Formal Encryption

The Abadi-Rogaway logic of formal encryption is simple to treat, but is complex enough to reveal many subtleties that might occur in a protocol. In this formalism an *expression* represents a multitude of messages that can be exchanged during a protocol. It can also be thought of as the data that an adversary has collected via observing a protocol. In this language all the expressions are built from keys and blocks of bits via pairing and encryption. We will start by presenting the original definitions introduced in [2]. Later, in Sections 4, 5 and 6, we will extend the AR definition of equivalence so that we can deal with different notions of security.

Definition 2.1. Let **Keys** $= \{K_1, K_2, K_3, ...\}$ be an infinite discrete set of symbols and **Blocks** $\subseteq \{0,1\}^*$ a nonempty subset. We define the *set of expressions*, **Exp**, by the grammar:

$$\mathbf{Exp} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Exp}, \mathbf{Exp}) \mid \mathbf{Enc}$$
$$\mathbf{Enc} ::= \{\mathbf{Exp}\}_{\mathbf{Keys}}$$

We will denote by *Keys*(M) the set of all keys occurring in M. We define the *set of subexpressions* of an expression M, *sub*(M), as the smallest subset of expressions containing M such that:

- $(M_1, M_2) \in \mathit{sub}(M) \implies M_1 \in \mathit{sub}(M)$ and $M_2 \in \mathit{sub}(M)$, and
- $\{M'\}_K \in \mathit{sub}(M) \implies M' \in \mathit{sub}(M)$.

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

We say that a key K *encrypts* an expression N in M if there is an expression N', such that $N \sqsubseteq N'$ and $\{N'\}_K \sqsubseteq M$. This induces a binary relation $\prec^M$ on *Keys*(M), that is, $K \prec^M K'$ iff K' encrypts K in M. We say that a subset $\mathcal{S}$ of *Keys*(M) is *cyclic* in M if the restriction of $\prec^M$ onto S is cyclic.

The reader should be aware that there are several different notions of cyclicity. According to our definition, expressions such as $\{\{M\}_K\}_K$ are not considered cyclic.

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

We define the *set of visible subexpressions* of an expression M, *vis*(M), as the smallest subset of expressions containing M such that:

- $(M_1, M_2) \in \mathit{vis}(M) \implies M_1 \in \mathit{vis}(M)$ and $M_2 \in \mathit{vis}(M)$, and
- $\{M'\}_K$ and $K \in \mathit{vis}(M) \implies M' \in \mathit{vis}(M)$.

We are now ready to define the set of *recoverable keys of* an expression M. The recoverable keys are those that an adversary can recover by looking at an expression. We define it as *R-Keys*$(M) = \mathit{vis}(M) \cap \mathit{Keys}(M)$. For more details, we refer to the example below, and [2].

We say that an encryption term $\{M'\}_K \sqsubseteq M$ is *undecryptable* in M if $K \notin$ *R-Keys*(M). Among the non-recoverable keys of an expression M, there is an important subset denoted by *B-Keys*(M). The set *B-Keys*(M) contains those keys which encrypt the outermost undecryptable terms. Formally, for an expression M, we define *B-Keys*(M) as

$$\textit{B-Keys}(M) = \left\{ \begin{matrix} K \in \mathit{Keys}(M) \mid \{M\}_K \in \mathit{vis}(M) \\ \text{but } K \notin \textit{R-Keys}(M) \end{matrix} \right\}$$

Example 2.2. Let M be the following expression

$$((\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}), ((K_2, \{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}), \{K_5\}_{K_2})).$$

In this case, *Keys*$(M) = \{K_1, K_2, K_3, K_4, K_5, K_6, K_7\}$. It is of course not necessary to use the first 7 keys; we could have used others. The set of recoverable keys is *R-Keys*$(M) = \{K_2, K_5, K_6\}$, because an adversary sees the non-encrypted K_2, and with that he can decrypt $\{K_5\}_{K_2}$, hence recovering K_5; then, decrypting twice with K_5, K_6 can be revealed. We also have that *B-Keys*$(M) = \{K_3, K_4\}$.

Abadi and Rogaway also defined an equivalence notion on the set of expressions. This equivalence expresses the fact that an adversary cannot distinguish certain messages. Abadi and Rogaway introduced this notion assuming that an adversary cannot distinguish any two undecryptable formal ciphers. However, if we want to express the fact that some partial information may be revealed about the plaintext or about the encrypting key, we need to adjust the definition of the equivalence. We first discuss two specific cases, the so called which-key revealing encryption schemes and the One-Time Pad (Sections 4 and 5), and adjust the equivalence-notion in the formal case, and then give a general treatment of such adjustments (Section 6).

3. Computational Model

In the Computational Model of Cryptography each message is a sequence of bits so, let **strings** $= \{0,1\}^*$. In order to be able to build up longer messages from shorter ones, we assume that an injective *pairing function* $[\cdot,\cdot]$: **strings** $\times$ **strings** $\to$ **strings** is given. Let **plaintexts**, **ciphertexts** and **keys** be nonempty subsets of **strings** and let **0** be a fixed particular element in **plaintexts**.

Definition 3.1 (Encryption Scheme). A *computational symmetric encryption scheme* is a tripple $\Pi = (\mathcal{K}, E, D)$ where

- $\mathcal{K}$: **parameters** $\times$ **coins** $\rightarrow$ **keys** is a key-generation algorithm with security parameter $\eta \in$ **parameters** $= \mathbb{N}$;
- E : **keys** $\times$ **strings** $\times$ **coins** $\rightarrow$ **ciphertexts** is an encryption function;
- D : **keys** $\times$ **strings** $\rightarrow$ **plaintexts** is such that for all $k \in$ **keys** and $\omega \in$ **coins**,

$$\begin{aligned} D_k(E_k(m,\omega)) = m \quad & \text{for all } m \in \textbf{plaintexts}, \\ D_k(E_k(m',\omega)) = \mathbf{0} \quad & \text{for all } m' \notin \textbf{plaintexts}. \end{aligned}$$

All of $\mathcal{K}$, E and D are computable in polynomial time in the size of the input, not counting the coins.

Computational Equivalence. In the computational setting, we assume that an adversary has access to computers with limited computational power. The notion 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 ciphertexts apart. Namely, messages are in fact ensembles (because of the security parameter) of random variables (since key generation and encryption are random), 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 notions of *negligible function* and *computational indistinguishability*:

Definition 3.2. A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ 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 $\eta \geq n_c$.

Definition 3.3. Let F_η and G_η (for $\eta \in$ **parameters**) be two sequences of random variables taking values in **strings**. 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 indistinguishable*, if for any probabilistic polynomial-time adversary $\mathcal{A}_\eta$,

$$\Pr\Big[x \xleftarrow{R} \mathit{Dist}(F_\eta) : \mathcal{A}_\eta(x) = 1\Big] - \Pr\Big[x \xleftarrow{R} \mathit{Dist}(G_\eta) : \mathcal{A}_\eta(x) = 1\Big]$$

is a negligible function of η.

Now that we know the meaning of indistinguishability of two ensembles, we can define security as the incapacity for an adversary to distinguish certain *encryption oracles*. For any key $k \in$ **keys**, an encryption oracle $\mathcal{E}_k(\cdot)$ is an algorithm that, on an input $x \in$ **strings**, outputs $E_k(x)$. The oracle $\mathcal{E}_k(\mathbf{0})$, on an input $x \in$ **strings**, outputs $E_k(\mathbf{0})$ In their seminal paper, AR defined several different notions of security. They defined type-0 security as follows:

Definition 3.4. We say that a computational encryption scheme is type-0 secure, if no probabilistic polynomial-time adversary can distinguish the pair of oracles $(\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot))$ from the pair of oracles $(\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0}))$ 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 $(\mathcal{E}_k(\mathbf{0}), \mathcal{E}_k(\mathbf{0}))$,

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

is a negligible function of η.

Intuitively the above formula says 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, the adversary can do all kinds of (probabilistic polynomial-time) computations including several queries to the oracles. He can even query the oracles with messages that depend on previously given answers of the oracles. We should remark that the keys used by the oracles for encryption do not change while the adversary queries the oracles. After this game, the adversary has to decide with which pair of oracles he was interacting. The adversary wins the game if he can decide for the correct one with a probability bigger than $\frac{1}{2}$, or equivalently if he can distinguish between the two. If this difference is negligible, as a function of η, 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 $\mathbf{0}$ instead, but he cannot tell either whether the encryptions by the pair were done with the same key, or with keys that had been separately generated. All the work presented by AR was done using this security level. It is possible to relax the power of the encryption scheme and by that we obtain several different notions of security.

Definition 3.5. 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.

Here, we do not require that encryption with different keys should not be possible to be detected, as was the case in the type-0 security. This notion is very convenient to discuss the expansion of the Abadi-Rogaway logic to match this case, and we therefore stick to it for the moment. Nevertheless, as we will see later, our work can be applied to several other types of security.

4. Soundness and Completeness for Type-2 Schemes

In order to prove any relation between the formal and computational worlds, we need to translate a formal expression to something that computational theory can handle. The translation, which we call interpretation results in a sequence of random variables (and their distributions) indexed by the security parameter. Namely, to each valid formal expression M and security parameter η, the interpretation assigns a random variable $\Phi_\eta(M)$ taking values in **strings**. Intuitively, the interpretation works as follows: blocks are interpreted as **strings**; each key is interpreted by running the key generation algorithm; pairs are translated into computational pairs and formal encryptions terms are interpreted by running the encryption algorithm.

AR presented such interpretation in an algorithmic way, which we include in the appendix. We will denote by $[\![M]\!]_{\Phi_\eta}$ the distribution of $\Phi_\eta(M)$ and by $[\![M]\!]_\Phi$ the ensemble of $\{[\![M]\!]_{\Phi_\eta}\}_{\eta\in\mathbb{N}}$.

4.1. Formal Equivalence, and Expansion of the Logic for Type-2 Schemes

Formal equivalence is meant to express that for an adversary, certain messages look the same. In the original AR treatment (that was based in type-0 security), any two encryption terms that were encrypted with non-recoverable keys looked the same to the adversary. For that, formal encryption terms encrypted with non-recoverable keys in an expression were replaced with a box, $\Box$, and if the resulting *pattern* agreed with the pattern of another expression up to *key-renaming*, then the two expressions were said to be equivalent. In a type-2 secure encryption scheme, an adversary may distinguish encryption terms that were encrypted with different keys, and therefore using the same box for all replacement will not work, the boxes have to be indexed by the encrypting keys. The patterns are hence defined in the following way:

$$\begin{aligned}\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \\ \mid \Box_{\mathbf{Keys}}\end{aligned}$$

The *pattern of an expression* for the type-2 case is defined as follows:

Definition 4.1. For an expression M, the pattern of M, *pattern*(M), is obtained from M by replacing each undecryptable term $\{M'\}_K \sqsubseteq M$ by $\Box_K$.

Definition 4.2. We say that two valid expressions M and N are *equivalent*, and denote it by $M \cong N$, if there is a *key-renaming function*, *i.e.*, a bijection $\sigma : \mathbf{Keys} \to \mathbf{Keys}$, such that *pattern*$(M)\sigma =$ *pattern*(N), where for any pattern Q, $Q\sigma$ denotes the pattern obtained from Q by replacing all occurrences of keys K in Q by $\sigma(K)$ (including those occurrences as indexes of $\Box$).

Example 4.3. Let N be the expression

$$\begin{aligned}((\{0\}_{K_8},\quad \{100\}_{K_1}\ ),((K_7, \\ \{(\{0101\}_{K_9},\{K_8\}_{K_5})\}_{K_5}),\{K_5\}_{K_7})).\end{aligned}$$

We have that $R\text{-}Keys(N) = \{K_5, K_7, K_8\}$, and so, in this case, *pattern*(N) is

$$((\{0\}_{K_8},\Box_{K_1}),((K_7,\{(\Box_{K_9},\{K_8\}_{K_5})\}_{K_5}),\{K_5\}_{K_7})).$$

Defining M as in Example 2.2, *pattern*(M) is

$$((\{0\}_{K_6},\Box_{K_4}),((K_2,\{(\Box_{K_3},\{K_6\}_{K_5})\}_{K_5}),\{K_5\}_{K_2})).$$

Now, if we replace $K_6 \to K_8$, $K_4 \to K_1$, $K_2 \to K_7$, $K_3 \to K_9$ and $K_5 \to K_5$ in M, the pattern of M turns into the pattern of N, so M and N are equivalent.

With these definitions, the following soundness and completeness theorems can be proved:

Theorem 4.4. Let M and N be expressions such that $B\text{-}Keys(M)$ and $B\text{-}Keys(N)$ are not cyclic in M and N respectively. Let Π be a type-2 secure encryption scheme. Then, $M \cong N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$.

In the other direction we have that, $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ implies $M \cong N$ for arbitrary expressions M and N if and only if the following conditions hold: for any $K, K', K'' \in \mathbf{Keys}$, $B \in \mathbf{Blocks}$, $M, M', N, N' \in \mathbf{Exp}$,
(i) no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$;
(ii) if $[\![(K,\{M\}_K)]\!]_\Phi \approx [\![(K'',\{M'\}_{K'})]\!]_\Phi$, then $K' = K''$;
(iii) if $[\![(\{M\}_K,\{M'\}_K)]\!]_\Phi \approx [\![(\{N\}_{K'},\{N'\}_{K''})]\!]_\Phi$ then $K' = K''$.

Let us now discuss the conditions in the completeness part above in some detail. Condition (i) requires that different types of objects, blocks, keys, pairs and encryption terms should be distinguishable to achieve completeness; this can be ensured by tagging each object with its type, as suggested in [2]. We call condition (ii) *weak confusion-freeness*. This condition is in fact equivalent to weak key-authenticity that was introduced by Horvitz and Gligor in [11] in the case of type-0 schemes; it essentially means that decrypting with the wrong key should be detectable in a probabilistic sense. Finally, condition (iii) requires that encryption with different keys should be detectable. The type-2 condition in Definition 3.5 *allows* that encrypting with different keys may be detectable, but it does not *require* it. That is good for soundness, but in order to achieve completeness for the formal equivalence we introduced, we need to assume that encryption with different keys is detectable. A purely computational condition that implies condition (iii) is to require that for some probabilistic polynomial-time algorithm $\mathcal{A}_\eta$,

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

is not a negligible function of η.

We can prove similar theorems for length revealing (type-1) encryption schemes and both which-key and length-revealing (type-3) encryption schemes. We do not state those theorems here since they will follow as corollaries of our general theorems.

5. Soundness and Completeness for One-Time Pad

Besides the computational, there are other possible important notions of indistinguishability. For example, we may say that two distributions are indistinguishable, if they are identical. We can still consider interpretations of formal expressions, and check soundness and completeness for this case. As an example, let us now consider a specific implementation of the One-Time Pad.

Let **strings** $:= \{0,1\}^*$ with the following pairing function: For any two strings $x, y \in$ **strings** we can define the pairing of x and y as $[x,y] := \langle x, y, 0, 1_{|y|}\rangle$ where $\langle\ ,\ ,\ ...\ ,\ \rangle$ 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 the 0 separates the strings from the 1's. Let **blocks** be those strings that end with 001. The ending is just a tag, it shows that the meaning of the string is a block.

Key-Generation. In case of the OTP, the length of the encrypting key must match the plaintext, hence we need a separate key-generation for each length. 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 $\mathbf{keys}_n := \{k \mid k \in \mathbf{strings},\ |k| = n,\ k \text{ ends with } 010\}$. Let $\mathbf{keys} := \bigcup_4^\infty \mathbf{keys}_n$. For $k \in \mathbf{keys}$, let $\mathrm{core}(k)$ denote the string that we get from k by cutting the tag 010.

Encryption. Let the domain of the encryption function, Dom_E, be those elements $(k, x) \in \mathbf{keys} \times \mathbf{strings}$, for which $|k| = |x| + 3$, and let $E_k(x) := \langle \mathrm{core}(k) \oplus x, 110\rangle$. The tag 110 informs us that the string is a ciphertext. Notice that this encryption is not probabilistic, $E_k(x)$ is not a random variable (or, in other words a constant random variable). Notice also, that the tag of the plaintext is not dropped, that part is also encrypted.

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 $k \oplus x$.

Indistinguishability. As we mentioned, let us now call two distributions indistinguishable, if they are identical.

5.1. Interpretation for One-Time Pad

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 2. Therefore, we have to expand the logic so that we can talk about the length of an expression.

Definition 5.1. We assume that some length function $l : \mathbf{Keys} \to \{4, 5, ...\}$ is given on the keys symbols. The length of a block 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 $\mathbf{Exp}$ by induction: $l((M,N)) := l(M) + 2l(N) + 1$, $l(\{M\}_K) := l(M) + 3$ if $l(M) = l(K) - 3$, and $l(\{M\}_K) = 0$ otherwise.

The *valid expressions* 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 is used twice to encrypt. This latter condition is necessary to prevent leaking information because of the properties of the OTP.

Definition 5.2. We define the *valid expressions for OTP* as $\mathbf{Exp}_{\mathrm{OTP}} = \{M \in \mathbf{Exp} \mid M' \sqsubseteq M$ implies $l(M') > 0$, and each key encrypts at most once in $M\}$.

The interpretation for the OTP is defined similarly to the type-2 case with some minor changes regarding the tagging of the messages, and there is no security parameter here, so the interpretation outputs one random variable only for each formal expression. For full details check the algorithm in the appendix.

5.2. Formal Equivalence and Expansion of the Logic for One-Time Pad

As in the case of type-2 encryption schemes, here we have to find also a suitable equivalence relation for the formal expressions. We now assign different boxes to encryption terms of different length. (We could use boxes indexed by the keys here too, see Example 6.24.) That is, we define the patterns as:

$$\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \\ \mid \Box_{\{4,5,\dots\}}$$

In the case of One-Time Pad, the patterns, and equivalence of expressions can be defined the following way:

Definition 5.3. For a valid expression M, the pattern of M, *pattern*(M), is obtained by replacing each undecryptable term $\{M'\}_K \sqsubseteq M$ by $\Box_{l(\{M'\}_K)}$, where $l(\{M'\}_K)$ denotes the formal length of $\{M'\}_K$ (which is in fact the same as $l(K)$).

Definition 5.4. We say that two expressions M and N are equivalent, and denote it by $M \cong_{\mathrm{OTP}} N$, if there exists a length-preserving key-renaming function such that *pattern*$(M)\sigma =$ *pattern*(N).

Then, the following soundness and completeness theorems can be proven:

Theorem 5.5. Let M and N be two valid expressions in $\mathbf{Exp}_{\mathrm{OTP}}$ such that *B-Keys*(M) and *B-Keys*(N) are not cyclic in M and N respectively. Then, $M \cong_{\mathrm{OTP}} N$ implies that $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$ are the same probability distributions.

Let M and N be two valid expressions in $\mathbf{Exp}_{\mathrm{OTP}}$. Then if $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$ have the same probability distributions, we have that $M \cong_{\mathrm{OTP}} N$.

In the completeness theorem for OTP we do not have any side condition as in Theorem 4.4. Note that here the analogue of the condition (i) from Theorem 4.4 is immediate due to the tagging. For (ii), the analogue also follows from the tagging since decrypting with the wrong key will result in a meaningless text. The analogue of (iii) is meaningless in this case since we just encrypt at most once with each key.

6. A General Treatment for Symmetric Encryption

We provide a general probabilistic framework for symmetric encryption, which contains both the computational and the information-theoretic description as special cases. Keys, plaintexts and ciphertexts are elements of some discrete set $\overline{\textbf{strings}}$. This is $(\{0,1\}^*)^\infty$ in the case of a computational treatment, and it is $\{0,1\}^*$ for the information-theoretic description. The elements of $(\{0,1\}^*)^\infty$ are sequences in $\{0,1\}^*$, corresponding to parametrization by the security parameter.

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 longer messages from shorter ones, 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. Key-generation is represented by a random variable $\mathcal{K} : \Omega_{\mathcal{K}} \to \overline{\textbf{keys}}$, over a discrete probability field $(\Omega_{\mathcal{K}}, \mathrm{Pr}_{\mathcal{K}})$. In a given scheme, more than one key-generation is allowed.

Encryption. 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 $(\Omega_E, \mathrm{Pr}_E)$. The values of this random variable are in $\overline{\textbf{strings}}$ and are denoted by $E_k(x)(\omega)$, whenever $\omega \in \Omega_E$.

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 satisfying $D_k\big(E_k(x)(\omega)\big) = x$ for all $\omega \in \Omega_E$ and $x \in \overline{\textbf{plaintext}}$.

The notion of *indistinguishability* is important both in case of computational and information-theoretic treatments of cryptography. It expresses when there is only very small probability to tell two probability distributions apart.

Indistinguishability. We assume that an equivalence relation called *indistinguishability* is defined on distributions over $\overline{\textbf{strings}}$. We will denote this relation by $\approx$. We will also say that two random variables taking values in $\overline{\textbf{strings}}$ are equivalent (indistinguishable) if (and only if) their distributions are equivalent; 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;
(iii) For random variables $F : \Omega_F \to \overline{\textbf{strings}}$ and $G : \Omega_G \to \overline{\textbf{strings}}$, if $F \approx G$, the followings must hold: If π^i denotes the projection onto one of the components of $\overline{\textbf{strings}} \times \overline{\textbf{strings}}$, then $\pi^i \circ [\cdot,\cdot]^{-1} \circ F \approx \pi^i \circ [\cdot,\cdot]^{-1} \circ G$ for $i = 1, 2$;
(iv) If $F' : \Omega_F \to \overline{\textbf{strings}}$, $G' : \Omega_G \to \overline{\textbf{strings}}$ 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 $\alpha, \beta : \overline{\textbf{strings}} \to \overline{\textbf{strings}}$ are 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 $F \approx G$.

Indistinguishability needs to satisfy some further properties under encryption and decryption that we will specify under the definition of encryption schemes below.

Example 6.1. The simplest example for indistinguishability is that it holds between two random variables if and only if their distributions are identical.

Example 6.2. The standard notion of computational indistinguishability in [21] is also a special case of the general definition. In this case $\overline{\textbf{strings}} = (\{0,1\}^*)^\infty = \textbf{strings}^\infty$. Random variables of computational interest have the form $F : \Omega_F \to \textbf{strings}^\infty$ and have independent components; *i.e.*, for $\eta \in \mathbb{N}$ security parameter, denoting the η'th component of F by $F_\eta : \Omega_F \to \textbf{strings}$, it is required that F_η and $F_{\eta'}$ are independent random variables for $\eta \neq \eta'$. Indistinguishability then is phrased with the ensemble of probability distributions of the components of the random variables.

Definition 6.3. An *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 ciphertexts encrypted by E, and $\approx$ is the indistinguishability defined above. We require that for any $i \in I$, the probability distribution of $\mathcal{K}_i$ be distinguishable from any constant in $\overline{\textbf{strings}}$, 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 $i, j \in I$. The indistinguishability relation $\approx$, besides satisfying the properties stated before, needs to be such that if F and G are random variables taking values in $\overline{\textbf{strings}}$, and $\mathcal{K}_i$ is a key-generation such that the distribution of $[\mathcal{K}_i, F]$ is indistinguishable from the 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 indistinguishable 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 indistinguishable random variables.
Here 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.

6.1. Equivalence of Expressions

In their treatment, Abadi and Rogaway defined equivalence of expressions via replacing encryption terms encrypted with non-recoverable keys in an expression by a box; two expressions then were declared equivalent if once these encryption terms were replaced, the received *patterns* looked the same up to *key-renaming*. This method implicitly assumes, that an adversary cannot distinguish any undecryptable terms. However, if we want to allow leaking of partial information, we need to modify the definition of equivalence.

Before introducing our notion of equivalence of expressions, we postulate an equivalence notion $\equiv_{\mathbf{K}}$ on the set of keys, and another equivalence, $\equiv_{\mathbf{C}}$ on the set of *valid* encryption terms. The word *valid*, defined precisely below, is meant for those encryption terms (and expressions) that "make sense". Then, the equivalence on the set of valid expressions will be defined with the help of $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$.

The reason for postulating equivalence on the set of keys is that we want to allow many key-generation processes in the probabilistic setting. We therefore have to be able to distinguish formal keys that were generated by different key-generation processes. Therefore, we assume that an equivalence relation $\equiv_{\mathbf{K}}$ is given on the set of keys such that each equivalence class contains infinitely many keys. Let $\mathcal{Q}_{\mathbf{Keys}} := \mathbf{Keys}\big/\equiv_{\mathbf{K}}$.

Definition 6.4. 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.

The set $\mathbf{Exp}$ is often too big to suit our purposes. For example, sometimes we require that certain messages can be encrypted with certain keys only. We therefore define the set of valid expressions:

Definition 6.5. 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 $sub(M) \subset \mathbf{Exp}_{\mathcal{V}}$ and any number of pairs of elements in $sub(M)$ are also in $\mathbf{Exp}_{\mathcal{V}}$; and
(iii) for any key-renaming function σ, $M \in \mathbf{Exp}_{\mathcal{V}}$ iff $M\sigma \in \mathbf{Exp}_{\mathcal{V}}$. Given a set of valid expressions, the set of *valid encryption terms* is $\mathbf{Enc}_{\mathcal{V}} := \mathbf{Enc} \cap \mathbf{Exp}_{\mathcal{V}}$.

Equivalence of valid expressions is meant to incorporate the notion of security into the model: we want two expressions to be equivalent when they look the same to an adversary. If we think that the encryption is so secure that no partial information is revealed, then all undecryptable terms should look the same to an adversary. If partial information, say repetition of the encrypting key, or length is revealed, then we have to adjust the notion of equivalence accordingly. We do this by introducing an equivalence relation on the set of valid encryption terms in order to capture which ciphertexts an adversary can and cannot distinguish; in other words, what partial information (length, key, etc...) can an adversary retrieve from the ciphertext.

Hence, from now on, we assume that there is an equivalence relation, $\equiv_{\mathbf{C}}$ given on the set of valid encryption terms, with the property that for any $M, N \in \mathbf{Enc}_{\mathcal{V}}$ and σ key-renaming function, $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$. Let $\mathcal{Q}_{\mathbf{Enc}} := \mathbf{Enc}_{\mathcal{V}} \big/ \equiv_{\mathbf{C}}$.

Since we required that $M \equiv_{\mathbf{C}} N \in \mathbf{Enc}_{\mathcal{V}}$, if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$ whenever σ is a key-renaming function, σ induces a renaming on $\mathcal{Q}_{\mathbf{Enc}}$, which we also denote by σ.

Example 6.6. We will consider encryption schemes where an adversary can recognize when two encryption terms were encrypted with different keys. For this case, we will need to define $\equiv_{\mathbf{C}}$ so that two encryption terms are equivalent if and only if they are encrypted with the same key.

Example 6.7. In [18], the authors find it useful to define a length-function on $\mathbf{Exp}$ by specifying $l(K) := 1$ for $K \in \mathbf{Keys}$, $l(B) := 1$ for $B \in \mathbf{Blocks}$, $l((M, N)) := l(M)+l(N)$, and $l(\{M\}_K) := l(M)+1$. Two encryption terms are then considered to be indistinguishable for an adversary if and only if they have the same length. In this case, we define $\equiv_{\mathbf{C}}$ so that it equates encryption terms with the same length, and hence an element of $\mathcal{Q}_{\mathbf{Enc}}$ will contain all encryption terms that have a specific length.

Definition 6.8. 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 $\mathbf{Keys}$, and $\equiv_{\mathbf{C}}$ is an equivalence relation on $\mathbf{Enc}_{\mathcal{V}}$; 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 $M, N \in \mathbf{Enc}_{\mathcal{V}}$, then $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$; and
(iii) replacing an encryption term within a valid expression with another equivalent valid encryption term results in a valid expression.

To define the equivalence of expressions, we first assign to each valid expression an element in the set of *patterns*, $\mathbf{Pat}$, defined the following way:

Definition 6.9. Let the set of patterns defined by the following grammar:

$$\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \\ \mid \square_{\mathcal{Q}_{\mathbf{Enc}}}$$

Definition 6.10. For a valid expression M, the pattern of M, *pattern*(M), is obtained by replacing each undecryptable term $\{M'\}_K \sqsubseteq M$ ($K \notin R\text{-}Keys(M)$) by $\square_{\mu(\{M'\}_K)}$, where $\mu(\{M'\}_K) \in \mathcal{Q}_{\mathbf{Enc}}$ denotes the equivalence class containing $\{M'\}_K$.

We say that two valid expressions M and N are *equivalent*, and denote it by $M \cong N$, if there is a key-renaming σ such that *pattern*$(M)\sigma =$ *pattern*(N), where for any pattern Q, $Q\sigma$ denotes the pattern obtained by renaming all the keys and the box-indexes (which are equivalence classes in $\mathcal{Q}_{\mathbf{Enc}}$) in Q with σ.

Example 6.11. In the case when the elements of $\mathcal{Q}_{\mathbf{Enc}}$ contain encryption terms encrypted with the same key, there is a one-to-one correspondence between $\mathcal{Q}_{\mathbf{Enc}}$ and $\mathbf{Keys}$, and therefore we can index the boxes with keys instead of the elements in $\mathcal{Q}_{\mathbf{Enc}}$: $\square_K$, $K \in \mathbf{Keys}$. Then if N is the same expression as in Example 4.3, the pattern according to the above definition is the same as we had in that example. M and N there are equivalent according to our definition of equivalence above.

6.1.1. Proper Equivalence of Ciphers

In order to make the soundness and completeness proofs work, we need to have some restrictions on $\equiv_{\mathbf{C}}$; without any restrictions, the proofs will never work. The condition that we found the most natural for our purposes is what we called *proper equivalence*, defined below. This condition will make soundness work. For completeness, besides proper equivalence, we need to assume something for the relationship of $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$. We call our assumption *independence*, and it is defined in Definition 6.17.

Definition 6.12. We say that an equivalence relation $\equiv_{\mathbf{C}}$ on $\mathbf{Enc}_{\mathcal{V}}$ is *proper*, if for any finite set of keys S, if $\mu \in \mathcal{Q}_{\mathbf{Enc}}$ 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 6.13. The equivalence $\equiv_{\mathbf{C}}$ of Example 6.6 and of Example 6.7 are both proper.

The following propositions that we present here are needed for proving our general soundness and completeness results. In order to be able to state them, for each $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, we introduce the set $\mu_{\text{key}} := \{K \in \mathbf{Keys} \mid \text{there is a valid expression } M \text{ such that } \{M\}_K \in \mu\}$. Full proofs can be found in [4]

Proposition 6.14. 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 $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, μ_{key} has either one, or infinitely many elements.

Proposition 6.15. 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 $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, $|\mu_{\text{key}}| = |\sigma(\mu)_{\text{key}}|$.

The most important proposition about properness is the following:

Proposition 6.16. 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 encryption terms, 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) 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 key K, 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 $K \not\sqsubseteq C_\nu$, $K \not\sqsubseteq C_{\nu'}$.

Given sets $\mathfrak{C}$ and S as in the conditions of the proposition, let $\mathfrak{R}(\mathfrak{C}, S)$ denote the nonempty set $\mathfrak{R}(\mathfrak{C}, S) := \{\{C_\nu\}_{\nu \in \mu(\mathfrak{C})} \mid C_\nu \in \nu, \text{ and } \{C_\nu\}_{\nu \in \mathfrak{C}} \text{ and } S \text{ satisfy conditions (i), (ii), (iii) of Proposition 6.16}\}$.

Another useful property, satisfied in our applications, and that we will need for the completeness result, is the following:

Definition 6.17. 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 encryption terms 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 $C \in \mathfrak{C}$, $C\sigma \equiv_{\mathbf{C}} C\sigma'$.

Example 6.18. The trivial $\equiv_{\mathbf{K}}$ equating all keys and the equivalence $\equiv_{\mathbf{C}}$ of Example 6.6 and of Example 6.7 are independent in both cases.

6.2. Interpretation

The idea of the interpretation is to describe messages that are built from blocks of strings and keys via pairing and encryption. To each valid formal expression M, the interpretation assigns a random variable $\Phi(M)$ taking values in $\overline{\mathbf{strings}}$. We do not give one specific interpreting function though, we will just say that a function Φ is an interpretation if it satisfies certain properties. We assume, that a function ϕ is fixed in advance, which assigns to each formal key a key-generation algorithm. If $\Phi(B) \in \overline{\mathbf{strings}}$ (constant random variable) 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 interpretations 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 interpretation 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. The precise definition is quite technical and we included that in the Appendix. Here we try to make it clear via an example:

Example 6.19. For $M = ((\{0\}_{K_{10}}, K_5), \{K_{10}\}_{K_5})$, the interpretation is $\Phi(M) : (\Omega_E \times \Omega_E) \times (\Omega_{\phi(K_5)} \times \Omega_{\phi(K_{10})}) \to \overline{\mathbf{strings}}$, $\Phi(M)(\omega_1, \omega_2, \omega_3, \omega_4) = [[E_{\phi(K_{10})(\omega_4)}(\Phi(0))(\omega_1), \phi(K_5)(\omega_3)], \quad E_{\phi(K_5)(\omega_3)}(\phi(K_{10})(\omega_4))(\omega_2)]$. There are four instances of randomness, two coming from the generating a key twice (for K_5 and for K_{10}), and encrypting twice.

6.3. Soundness and Completeness

An interpretation assigns a random variable $\Phi(M)$ (and the distribution $[\![M]\!]_\Phi$ of $\Phi(M)$) to a formal valid expression M. On the set of valid expressions the equivalence $\cong$ equates expressions that a formal adversary supposedly cannot distinguish, whereas the equivalence $\approx$ equates random variables (and distributions) that a probabilistic adversary is not supposed to be able to distinguish. The question is, how the formal and the probabilistic equivalence are related through the interpretation. We say that soundness holds if $M \cong N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$, whereas we say that completeness holds if $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ implies $M \cong N$.

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 $\mathcal{Q}_{\mathbf{Enc}}$. It is clear that in the extreme case, when the equivalence on encryption terms, $\equiv_{\mathbf{C}}$, is defined so that two encryption terms 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 encryption terms 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 $M \cong M'$ implies $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$ for certain specified pairs M, M', then $M \cong N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ for any two pairs of valid expressions M, N (with certain acyclicity).

For the definition of $\mathfrak{R}(\mathfrak{C}, S)$, see Section 6.1.1.

Theorem 6.20. Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be a formal logic for symmetric encryption such that for each $M \in \mathbf{Exp}_{\mathcal{V}}$, *B-Keys*$(M)$ is not cyclic in M. Assume that $\equiv_{\mathbf{C}}$ is proper. Let $\Pi = (\{\mathcal{K}_i\}_{i\in I}, E, D, \approx)$ be a general encryption scheme, Φ an interpretation of $\mathbf{Exp}_{\mathcal{V}}$ in Π. Then the following conditions are equivalent:

(i) Soundness holds for Φ: $M \cong N$, implies $\Phi(M) \approx \Phi(N)$.

(ii) For any $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ set of valid encryption terms, 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 followings hold: **if** $\left\{\{N_{i_j}\}_K\right\}_{j=1}^l \subset \mathfrak{C}$ and $M \in \mathbf{Exp}_{\mathcal{V}}$ are such that **(1)** $\{N_{i_1}\}_K, \{N_{i_2}\}_K, ..., \{N_{i_l}\}_K \sqsubseteq M$, **(2)** *R-Keys*$(M) \subset S$, **(3)** K does not occur anywhere else in M, **and if** we denote by M' the expression obtained by replacing in M each $\{N_{i_j}\}_K$ with $C_{\mu(\{N_{i_j}\}_K)}$, **then** $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$.

The proof of this theorem is motivated by the soundness proof in [2]. Full proof can be found in [4].

The idea of the proof is the following: Starting from two acyclic expressions $M_0 = M \cong N = N_0$, we create expressions $M_1, ..., M_b$ and $N_1, ..., N_{b'}$ such that M_{i+1} is received from M_i via a replacement of encryption terms as described in condition (ii). Acyclicity ensures that the encrypting key of the replaced encryption terms will not occur anywhere else. Similarly form N_{i+1} and N_i. We do this so that M_b and $N_{b'}$ will differ only in key renaming. Then, by condition (ii), $[\![M_{i+1}]\!]_\Phi \approx [\![M_i]\!]_\Phi$, and $[\![N_{i+1}]\!]_\Phi \approx [\![N_i]\!]_\Phi$. But, $[\![M_b]\!]_\Phi = [\![N_{b'}]\!]_\Phi$, and therefore the theorem follows.

Remark 6.21. The paper of Laud [13] addresses the possibility of getting rid of the acyclicity assumption. In order to obtain soundness for expressions with cycles, he leaves undecryptable terms that are encrypted by keys in cycles untouched (*i.e.* he does not replace these encryption terms with boxes). We could have proceeded the same way in our treatment as well. However, as Laud points it out, not replacing those encryption terms with boxes means that the adversary can decrypt them, which is not a reasonable assumption in general, therefore we included the acyclicity assumption.

Example 6.22. The soundness theorem we presented earlier for type-2 encryption schemes is a special case of the theorem above. In this case $\mathbf{Exp}_{\mathcal{V}} = \mathbf{Exp}$; the equivalence relation $\equiv_{\mathbf{C}}$ is as in Example 6.6, which is proper as we mentioned in Example 6.13; the equivalence relation $\equiv_{\mathbf{K}}$ is trivial here, all keys are equivalent. The elements $\mu \in \mathcal{Q}_{\mathbf{Enc}}$ are in one-to-one correspondence with the keys, so we can say $\mathcal{Q}_{\mathbf{Enc}} \equiv \mathbf{Keys}$, and thus the boxes are labeled with keys. Φ here gives an interpretation in the computational setting. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii) of the theorem, we can take $C_{L_i} := \{\mathbf{0}\}_{L_i}$, and then condition (ii) is satisfied, because the following proposition holds (full proof can be found in [4]):

Proposition 6.23. Consider an expression M, and a key $L \in$ *Keys*(M). Suppose that for some expressions $M_1, M_2, ..., M_l \in \mathbf{Exp}$, $\{M_1\}_L, \{M_2\}_L, ..., \{M_l\}_L \sqsubseteq M$, and assume also that L does not occur anywhere else in M. Then, denoting by M' the expression that we get from M by replacing each of $\{M_i\}_L$ that are not contained in any of M_j $(j \neq i)$ by $\{\mathbf{0}\}_L$, $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$ holds.

Hence, condition (ii) of the general soundness theorem is satisfied, so soundness holds for the type-2 case.

Example 6.24. Here we indicate that there is a formal logic for symmetric encryption such that we receive soundness as a special case of the above theorem when interpreting it in the One-Time Pad implementation presented in section 5. The formal equivalence we introduced for One-Time Pad in section 5 derives from taking the equivalence on encryption terms according to their length. However, the soundness part of theorem 5.5 then will not be a special case of our general theorem. So let us instead define $\equiv_{\mathbf{C}}$ so that two encryption terms are equivalent, iff (again) the encryption terms have the same encrypting key. The equivalence of keys, $\equiv_{\mathbf{K}}$ is defined with the help of a length-function l on the keys: two keys are equivalent iff they have the same length. Then the boxes will again be indexed by the encrypting keys. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii), take $C_{L_i} := \{0_{l(L_i)-3}\}_{L_i}$ (where $0_{l(L_i)-3}$ means $l(L_i) - 3$ many 0's). It is not hard to check that within this setting, condition (ii) of the soundness theorem is satisfied.

Example 6.25. For a discussion on type-1 schemes, recall now Example 6.7, where we cited the length-function Micciancio and Warinschi used in [18]. 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. The equivalence of encryption terms, $\equiv_{\mathbf{C}}$, for type-1 case is defined so that equivalence holds iff the formal length of the encryption terms are the same. This gives a proper equivalence. It is not very hard to see that condition (ii) of our general soundness theorem is satisfied.

It is clear that in order to be able to define equivalence on encryption terms 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 an encryption term will also vary then.

If the encryption scheme is such that for a fixed security parameter the size of the ciphertext depends only on the size of the plaintext, then it is possible to introduce a length-function on formal expressions that assign a sequence of length to each expression, each element of the sequence corresponding to a value of the security parameter. This length function again defines an equivalence relation, the boxes can be indexed by the sequences, and if the length function was chosen well, then soundness will follow.

Another way of dealing with length is to index the boxes with the type-tree of the replaced encryption term (*i.e.* two encryption terms are equivalent if their type-trees are identical) as Herzog did in [9].

Example 6.26. For type-3 encryption schemes, equivalence on encryption terms are defined so that equivalence holds iff the *encrypting keys and the lengths of the encryption terms* agree; this is a proper equivalence. Then, again, condition (ii) of the general theorem holds.

We finally present our completeness result. Condition (ii) is equivalent to what the authors in [11] call weak key-authenticity. A full proof of this theorem is available in [4].

Theorem 6.27. 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, encryption terms are distinguishable,
(ii) if $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$,
(iii) For any two pairs of valid encryption terms: $\{\{M_i\}_{L_i}\}_{i=1}^2$ and $\{\{N_i\}_{L'_i}\}_{i=1}^2$, from $[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_{L'_1}, \{N_2\}_{L'_2})]\!]_\Phi$ it follows that $(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong (\{N_1\}_{L'_1}, \{N_2\}_{L'_2})$.

The proof consists of two separate parts. In the first, it is shown that conditions (i) and (ii) imply that if M and N are valid expressions and $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$, then there is a key-renaming σ, such that apart from the boxes, everything else in the patterns of M and $N\sigma$ are the same, and the boxes in the two patterns must be in the same positions. Moreover, condition (iii) implies that picking any two boxes of the pattern of $N\sigma$, there is a key-renaming σ_1 such that applying it to the indexes of these boxes, we obtain the corresponding boxes in the pattern of M. Then the theorem follows, if we prove that using these pairwise equivalences of the boxes, we can construct a σ' that leaves the keys of $N\sigma$ outside the boxes untouched, and it maps the indexes of all the boxes of $N\sigma$ into the indexes of the boxes of M.

Remark 6.28. Observe, that condition (iii) of the theorem is trivially satisfied when there is only one box, that is, when all encryption terms 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 6.29. The completeness part of our earlier theorem for type-2 encryption schemes is clearly a special case of this theorem, because the formal language we introduced for type-2 schemes is such that $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent.

Example 6.30. The formal logic for OTP that we presented in Example 6.24 is such that $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent. Furthermore, condition (i) of Theorem 6.27 is satisfied due to the tagging we presented in Section 5. Condition (ii) is also satisfied because of the tagging: the reason ultimately is that decrypting with the wrong key will sometimes result invalid endings. Condition (iii) is also satisfied, since the pairs of encryption terms 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 encryption terms must be the same: $l(\{M_1\}_{L_1}) = l(\{N_1\}_{L'_1})$ and $l(\{M_2\}_{L_2}) = l(\{N_2\}_{L'_2})$ which implies $(\Box_{l(\{M_1\}_{L_1})}, \Box_{l(\{M_2\}_{L_2})}) = (\Box_{l(\{N_1\}_{L'_1})}, \Box_{l(\{N_2\}_{L'_2})})$. Therefore, $(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong (\{N_1\}_{L'_1}, \{N_2\}_{L'_2})$. In conclusion, the formal logic introduced in Example 6.24 is complete.

Example 6.31. In case of type-1 encryption schemes, 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 encryption scheme 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 encryption term.

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 of them, that is, when it is strictly which-key revealing and strictly length revealing.

7. Conclusions and Further Work

We have studied expansions of the Abadi-Rogaway logic of indistinguishability of formal cryptographic expressions. We have showed that, at least in the case of symmetric encryption, subtle distinctions among security levels of computational or information-theoretic encryption schemes can be faithfully reflected in the formal symbolic setting. We have introduced a general probabilistic framework, which includes both the computational and the information-theoretic encryption schemes as special cases. We have established soundness and completeness theorems in this general framework, as well as new applications to specific settings: an information-theoretic interpretation of formal expressions in One-Time Pad, and also computational interpretations in type-2 (which-key revealing) and type-3 (which-key and length revealing) encryption schemes based on computational complexity.

Because our theorems apply to weak encryption schemes, they also apply to strong, *e.g.,* CCA2-secure encryption schemes. However, the chosen ciphertext attacks or attacks exploiting malleability lie outside of the Abadi-Rogaway formal setting because its message space is rather parsimonious. We are exploring various expansions of the formal setting that would allow certain operations on bit strings such as *xor*, pseudorandom permutations, or exponentiation, in order to extend our soundness and completeness techniques to such richer formal settings. In particular, the definition of patterns appears to be rather subtle in such richer settings. We would also like to understand how our methods fit with the methods of [16].

We are also considering analogs of our results for asymmetric encryption. We do not foresee major obstacles in this direction. We also plan to extend our methods and investigate formal treatment of other cryptographic primitives. It would be interesting to see if our methods could be combined with the methods of [3, 5].

The problems related to cyclicity of keys, which lie beyond the scope of this paper, also deserve our attention. We are addressing these problems in our current work with Jonathan Herzog, in preparation.

References

[1] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, volume 2215 of *LNCS*, pages 82–94, Sendai, Japan, 2001. Springer.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002. Preliminary version presented at IFIP TCS 2000.

[3] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS)*, pages 220–230, Washington D.C., USA, 2003. ACM Press. Full version available at IACR ePrint Archive, Report 2003/015, January 2003.

[4] G. Bana. *Soundness and Completeness of Formal Logics of Symmetric Encryption.* PhD the-

sis, University of Pennsylvania, 2004. Available at www.math.upenn.edu/∼bana/banaphdthesis.pdf. Also available at IACR ePrint Archive.

[5] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, Las Vegas, NV, USA, 2001. IEEE Computer Society. Full version available at IACR ePrint Archive, Report 2000/067.

[6] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP)*, volume 3444 of *LNCS*, pages 157–171, Edinburgh, UK, 2005. Springer.

[7] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. Preliminary version presented at FOCS'81.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28(2):270–299, 1984. Preliminary version presented at STOC'82.

[9] J. Herzog. *Computational Soundness for Standard Assumptions of Formal Cryptography.* PhD thesis, Massachussets Institute of Technology, 2004. Available at http://theory.lcs.mit.edu/∼jherzog/papers/herzog-phd.pdf.

[10] J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 548–564, Santa Barbara, CA, USA, 2003. Springer.

[11] O. Horvitz and V. Gligor. Weak key authenticity and the computational completeness of formal encryption. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 530–547, Santa Barbara, CA, USA, 2003. Springer.

[12] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *Proc. 14th European Symposium on Programming (ESOP)*, volume 3444 of *LNCS*, pages 172–185, Edinburgh, UK, 2005. Springer.

[13] P. Laud. Encryption cycles and two views of cryptography. In *Proc. 7th Nordic Workshop on Secure IT Systems*, number 31, pages 85–100, Karlstad, Sweden, 2002. Karlstad University Studies.

[14] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology (ICISC)*, volume 2971 of *LNCS*, pages 55–66, Seoul, Korea, 2003. Springer.

[15] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polynomial-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security (CCS)*, pages 112–121, San Francisco, CA, USA, 1998. ACM Press.

[16] U. Maurer. Indistinguishability of random systems. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 110–132, Amsterdam, The Netherlands, 2002. Springer.

[17] D. Micciancio and S. Panjwani. Adaptive security of symbolic encryption. In *Proc. 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 169–187, Cambridge, MA, USA, 2005. Springer.

[18] 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.

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

[20] B. Warinschi. A computational analysis of the Needham-Schroeder protocol. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW)*, pages 248–262, Pacific Grove, CA, USA, 2003. IEEE Computer Society.

[21] A. C. Yao. Theory and applications of trapdoor functions. In *23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, Chicago, IL, USA, 1982. IEEE Computer Society.

A. Appendix

Algorithmic Interpretation of Expressions for Type-2 Systems

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**
    $x \xleftarrow{R}$ CONVERT(Q_1)
    $y \xleftarrow{R} E_{\tau(K)}(x)$
    return y

Algorithmic Interpretation of Expressions for One-Time Pad

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

return y

algorithm $\text{CONVERT}_{\text{OTP}}(N)$
  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 $[\text{CONVERT}_{\text{OTP}}(N_1),$
      $\text{CONVERT}_{\text{OTP}}(N_2)]$
  if $N = \{N_1\}_K$ **then**
    return
      $\langle E_{\tau(K)}(\text{CONVERT}_{\text{OTP}}(N_1)), 110\rangle$

Definition A.1 (Interpretation of Formal Expressions). Let $\Pi = (\{\mathcal{K}_i\}_{i\in I}, E, D, \approx)$ 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 recursively as

$$(\Omega_K, \Pr_K) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \text{ for } K \in \mathbf{Keys};$$
$$(\Omega_B, \Pr_B) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \text{ for } B \in \mathbf{Blocks};$$
$$(\Omega_{(M,N)}, \Pr_{(M,N)}) := (\Omega_M \times \Omega_N, \Pr_M \otimes \Pr_N);$$
$$(\Omega_{\{M\}_K}, \Pr_{\{M\}_K}) := (\Omega_E \times \Omega_M, \Pr_E \otimes \Pr_M).$$

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 $\phi : \mathbf{Keys} \to \{\mathcal{K}_i\}_{i\in I}$ is given assigning key generations to abstract keys, such that $\phi(K) = \phi(K')$ if and only if $K \equiv_{\mathbf{K}} K'$. Let $\iota : \{1, .., |Keys(M)|\} \to Keys(M)$ be a bijection enumerating the keys in $Keys(M)$. Let

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

The function $(M, M') \mapsto (\Phi_M(M') : \Omega_{M'} \times \Omega_{Keys(M)} \to \overline{\mathbf{strings}})$ defined whenever $M' \sqsubseteq M$, is called *an interpreting function*, if it satisfies the following properties:

$\Phi_M(B)(\omega_0, \omega) = \Phi_N(B)(\omega_0, \omega')$ for all M, N valid expressions, $B \in$ **Blocks**, $B \sqsubseteq M$, $B \sqsubseteq N$, and arbitrary $\omega \in \Omega_{Keys(M)}$, $\omega' \in \Omega_{Keys(N)}$. Let $\Phi(B) := \Phi_M(B)$.

$\Phi_M(K)(\omega_0, (\omega_1, ..., \omega_{|Keys(M)|})) = \phi(K)(\omega_{\iota^{-1}(K)})$ for $K \in Keys(M)$, with $\omega_j \in \Omega_{\phi(\iota(j))}$.

$\Phi_M((M', M''))((\omega', \omega''), \omega) =$
$= [\Phi_M(M')(\omega', \omega), \Phi_M(M'')(\omega'', \omega)]$ for all $\omega' \in \Omega_{M'}$, $\omega'' \in \Omega_{M''}$, and $\omega \in \Omega_{Keys(M)}$ if $(M', M'') \sqsubseteq M$.

$\Phi_M(\{M'\}_K)((\omega_E, \omega'), \omega) =$
$= E_{\Phi_M(K)(\omega_0,\omega)}(\Phi_M(M')(\omega', \omega))(\omega_E)$ for all $\omega_E \in \Omega_E$, $\omega' \in \Omega_{M'}$, $\omega \in \Omega_{Keys(M)}$ if $\{M'\}_K \sqsubseteq M$.

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

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).