Approximating Imperfect Cryptography in a Formal Model

Angelo Troina [1]

Dipartimento di Informatica, Università di Pisa, Italy

Alessandro Aldini [2]

Istituto STI, Università Carlo Bo, Urbino, Italy

Roberto Gorrieri [3]

Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy

Abstract

We present a formal view of cryptography that overcomes the usual assumptions of formal models for reasoning about security of computer systems, i.e. perfect cryptography and Dolev-Yao adversary model. In our framework, equivalence among formal cryptographic expressions is parameterized by a computational adversary that may exploit weaknesses of the cryptosystem to cryptanalyze ciphertext with a certain probability of success. To validate our approach, we show that in the restricted setting of ideal cryptosystems, for which the probability of guessing information that the Dolev-Yao adversary cannot derive is negligible, the computational adversary is limited to the allowed behaviors of the Dolev-Yao adversary.

1 Introduction

The use of formal methods for modeling and analyzing cryptographic operations is well-established. Since the seminal paper by Dolev and Yao [10] introduced a simple and intuitive formalization of cryptographic operations, many alternative definitions have been proposed on the basis of several approaches, ranging from modal logics to process algebras (see, e.g., [8,17,14,12,19,18,11]). Key to success of such a theory was the very simple idea behind the definition of ciphertext, which is based on the assumption of perfect cryptography.

[1] Email: troina@di.unipi.it
[2] Email: aldini@sti.uniurb.it
[3] Email: gorrieri@cs.unibo.it

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Simply put, a message encrypted with a given key K can be decrypted if and only if K is known, while in each other case such a message is a black box. More formally, $\{M\}_K$ (representing the encryption of M with the key K) and $\otimes$ (representing an undecryptable ciphertext) are always equivalent if K is not known. On the basis of such an assumption, an adversary can (i) decrypt ciphered information if and only if the needed key is known, (ii) capture plaintext, and (iii) encrypt plaintext with a known key. However, a real computational adversary is an arbitrary algorithm that collects large amount of ciphertext, exploits partial knowledge of information contained in a ciphertext, and performs exhaustive searches in order to crack ciphertext, compute ciphertext from a plaintext without knowing the related key, and guess secret keys. For instance, in [5] it is shown that an improper use of the block cipher Skipjack allows the adversary to perform an attack that is faster than exhaustive search, thus increasing the probability of retrieving data ciphered with an unknown key. As another example, the Wired Equivalent Privacy protocol for wireless networks falls short of accomplishing its security goals [6], because of an improper use of the stream cipher RC4. The lesson we learnt is that design of secure protocols is difficult, even if the underlying cryptographic primitives are believed to be secure. These considerations are discordant with the usual assumptions made by formal models, which do not define security in terms of the probability of successful attacks. As a consequence, in practice formal proofs are not enough to guarantee system security or, at least, they need specific assumptions about encryption.

In this paper, we overcome the limitations mentioned above. In particular, we interpret the adversary as a probabilistic polynomial time process that may randomly guess data, perform statistical analysis of exchanged information, exploit keys weakness, use well-known attacks to the used ciphering algorithm, and employ partial information to reduce the range of exhaustive searches. For such a model of adversary the probability of illegally cryptanalyzing information from a particular ciphertext may be not negligible. In other words, we abandon the perfect cryptography assumption and we take into account encryption schemes that may be violated. To this purpose, we use a probabilistic estimation of the robustness of the cryptosystem to decide the equivalence between formal cryptographic expressions. More formally, we define a function parameterized by the initial knowledge of the adversary, whose outcome is strictly related to the considerations surveyed above. Such an outcome represents an estimation of the probability of obtaining useful information from a given ciphertext. Consider, for instance, the expressions $N = (\{M\}_K, \{K\}_{K'})$, expressing a pair of ciphertexts, the second one containing the key needed to decrypt the first one, and $(\{M\}_K, K)$, expressing a ciphertext and the related key. The two expressions may be equivalent if, e.g., K' is a very short key, the used algorithm is a stream cipher, and in the initial knowledge of the adversary there exists a large amount of data encrypted in the same way by re-using K', so that, in practice, the probability for such an adversary of retrieving M from N in polynomial time can be considered equal to 1. Obviously, when computing the probability of retrieving data, the knowledge of the adversary increases as it succeeds in obtaining new information. Therefore, the probabilistic estimation of the adversary power always depends on the current knowledge of such an adversary.

The notion of equivalence we adopt takes into consideration the computational power of the adversary in order to establish the indistinguishability among different cryptographic expressions. However, it is possible that the estimation of the adversary capability of retrieving data is not accurate. Moreover, sometimes in practice the adversary cannot distinguish two expressions that, instead, are not equivalent because of negligible differences. In essence, the distinction between sets of cryptographic expressions may be too strong. To this aim, we approximate the closure among cryptographic expressions by introducing an ε-tolerance, which allows those expressions that require almost the same effort to reveal information to be indistinguishable from the viewpoint of the computational adversary. For instance, expressions $\{M\}_K$ and $\{rubbish\}_K$ are indistinguishable (both represent undecryptable text) if a probabilistic adversary can infer with a negligible probability information about M or *rubbish* from the ciphertexts encrypted with the unknown key K. That means, by borrowing the same terminology used in [13], the encryption scheme is *ideal*. Such an example suggests that equivalence in the formal view implies indistinguishability in the computational view if ideal encryption is assumed. In other words, if the probability of retrieving data that the Dolev-Yao adversary cannot obtain is negligible, then the expressive power of the computational adversary is limited to the allowed behaviors of the Dolev-Yao adversary. This is, indeed, the result shown in [1], where the formal view and the computational view of cryptography are related by providing a computational motivation for a formal treatment of encryption. Similarly, as a result of this paper, we show that under the same assumption of ideal encryption, our notion of approximate indistinguishability is implied by a classical notion of equivalence inspired by the formal model of Dolev and Yao.

The rest of the paper is organized as follows. In Sect. 2 we show how we extended the Dolev-Yao formal model with probabilistic information used to estimate the probability of successful attacks conducted by probabilistic polynomial time adversaries. In this framework we introduce a notion of probabilistic equivalence among cryptographic expressions. In Sect. 3 we relax such an equivalence through an approximate notion of indistinguishability, which relates expressions that can be considered the same up to small differences. In Sect. 4 we present a soundness result showing that such an approximate relation is implied by the equivalence relation of the Dolev-Yao model in the case ideal encryption is assumed. Some related work and concluding remarks are discussed in Sect. 5 and in Sect. 6, respectively.

2 Probabilistic Equivalence

The basic elements of our formal model are inspired by the Dolev-Yao encryption setting defined by Abadi and Rogaway [1]. However, in our setting, two cryptographic expressions turn out to be probabilistically equivalent if they yield the same information obtained with the same probability even through cryptanalysis attempts. Therefore, we abandon the usual Dolev-Yao abstraction and we take into consideration cryptanalysis attacks.

2.1 Cryptographic Expressions and Probabilistic Adversaries

We start by introducing the formal expressions and the machinery needed to compute an estimation of the adversary capability of retrieving information from such expressions. We use **String** to denote a finite set of plaintext messages, i.e. the set of binary strings of a fixed length (ranged over by $m, n, \ldots$), **Keys** to denote a fixed, nonempty set of key symbols (ranged over by $K, K', K'', \ldots$ and $K_1, K_2, K_3, \ldots$), such that **Keys** and **String** are disjoint, and **Exp** to denote the set of *expressions* defined by the grammar:

$M, N ::=$	expressions
K	key (for $K \in \mathbf{Keys}$)
m	string (for $m \in \mathbf{String}$)
(M, N)	pair
$\{M\}_K$	encryption (for $K \in \mathbf{Keys}$)

Informally, (M, N) represents the pairing of M and N, and $\{M\}_K$ represents the encryption of M under K via a symmetric encryption algorithm. Pairing and encryption can be nested, like, e.g., in $(\{\{(m, K)\}_{K_1}\}_{K_2}, K_1)$.

The *entailment* relation $M \mapsto N$ says that N can be derived from M. Formally, such a relation is inductively defined as the least relation satisfying the following properties:

$$
\begin{array}{ll}
M \mapsto M & \\
M \mapsto N_1 \ \wedge \ M \mapsto N_2 & \Rightarrow M \mapsto (N_1, N_2) \\
M \mapsto (N_1, N_2) & \Rightarrow M \mapsto N_1 \ \wedge \ M \mapsto N_2 \\
M \mapsto N \ \wedge \ M \mapsto K & \Rightarrow M \mapsto \{N\}_K \\
M \mapsto \{N\}_K \ \wedge \ M \mapsto K & \Rightarrow M \mapsto N
\end{array}
$$

In essence, $M \mapsto N$ expresses what the adversary obtains from M without any prior knowledge of the keys used in M. For instance, $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \mapsto K_3$, and $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \mapsto \{K_1\}_{K_2}$, but $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \not\mapsto K_1$. We point out that the entailment relation models the expressive power of the classical Dolev-Yao adversary and includes all the operations that such an adversary can perform in order to construct ciphertexts or extract plaintexts.

We now extend the model of Dolev and Yao by taking into account the possibility for an adversary of obtaining information from an expression $\{M\}_K$ without knowing the key K. For our purpose, we define the notion of *probabilistic pattern* $P._p$, which represents an expression P that does not contain undecryptable blocks and is associated with a parameter $p \in]0,1]$. The parameter p models the probability of getting the plaintext contained in P. Formally, we define the set **pPat** of probabilistic patterns with the grammar:

$P._p, Q._p ::=$	probabilistic patterns
$K._p$	key (for $K \in$ **Keys**)
$m._p$	string (for $m \in$ **String**)
$(P._p, Q._p)._p$	pair
$p \in]0,1]$	

A probabilistic pattern associated to a ciphertext is obtained by substituting every ciphered block with the corresponding expression in clear associated with the probability of obtaining information about it. Given any computational adversary $\mathcal{A}$ (described by a probabilistic polynomial time algorithm) and the initial knowledge represented by expression G, the probabilistic pattern associated with expression $\{m\}_K$ is expressed in terms of the probability of obtaining information about m and is formally defined as $m.p_{dec}(\{m\}_K, G)$. Function $p_{dec}(\{m\}_K, G)$ returns the probability of obtaining useful information from the ciphertext $\{m\}_K$ by employing the initial knowledge G. More formally, a computational adversary $\mathcal{A}$ has a probability Pr at most equal to the value expressed by p_{dec} of retrieving m from $\{m\}_K$ by exploiting G:

$$\boxed{Pr[m \leftarrow \mathcal{A}(\{m\}_K, G)] \leq p_{dec}(\{m\}_K, G) \text{ for all } \mathcal{A}}$$

Note that the outcome of p_{dec} is a value strictly greater than 0, because, even if with small probability, an adversary could always try to randomly guess the key. Besides, the value of p_{dec} depends on the knowledge G exploited to conduct the cryptanalysis attempt.

Intuitively, we could figure out the adversary as an arbitrary algorithm, executing in probabilistic polynomial time, that makes computations on ciphered texts in order to get information about the contained plaintext (see, e.g., [13] for a detailed description of adversaries within a computational model). We point out that the classical Dolev-Yao adversary obtains m from $\{m\}_K$ if and only if K can be derived from G. In such a case, if $G \mapsto K$, then $p_{dec}(\{m\}_K, G) = 1$. On the other hand, in a classical computational model assuming ideal encryption scheme [13] or type-0 secure encryption scheme [1],

p_{dec} is a negligible function, as it turns out that the probability of guessing information that cannot be derived through the classical Dolev-Yao model of cryptography is negligible. In the following we will consider a formal definition of negligible function and we will show that if p_{dec} is negligible, then we obtain a soundness result stating that the expressive power of the computational adversary is limited by that of the Dolev-Yao adversary [1,13].

In a more general scenario, in this paper we assume that function p_{dec} provides an outcome that depends on many factors that can violate the ideal encryption scheme assumption, such as the expected robustness of key K, the particular ciphering algorithm, the information collected by $\mathcal{A}$. However, function p_{dec} is not sufficient to define the probability of decrypting a ciphered block. Consider, for example, the expression $(\{\{m\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K)$. What is the probability of getting information about m? A simple and immediate answer could be $p_{dec}(\{\{m\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{m\}_{K_1})$ [4], that is the probability of sequentially cracking the two keys K_2 and K_1. However, we observe that if K is a weak key, then information about K_1 and K_2 can be easily derived from $\{(K_1, K_2)\}_K$ and, as a consequence, the cryptanalysis of $\{\{m\}_{K_1}\}_{K_2}$ may be simplified. Hence, the probability of success may vary according to the strategy the adversary uses when trying to cryptanalyze an expression. Obviously, since we have to assume that the adversary follows the optimal attack, we always associate to a ciphered block the maximum probability of getting information about it. As a consequence, we analyze all the possible cryptanalysis paths that the adversary can follow. To this purpose, we employ some auxiliary structures and functions, which we informally introduce as follows.

Given an expression M and initial knowledge G, we denote by $\mathbf{pKeys}_M^G$ the set of pairs of the form $T._p$, where $T \subseteq \mathbf{Keys}$ is a set of keys that can be obtained from the expression M, and $p \in]0, 1]$ is the probability of cracking all the keys contained in T by following a particular strategy. By employing the values contained in the set $\mathbf{pKeys}_M^G$, we get the following information. On the one hand, for each set T of keys that can be obtained from M we compute $pGuess_M^G(T)$, which is the maximum probability of cracking all keys in T. On the other hand, we compute a parameter $pMax_M^G$, associated to the expression M, expressing the maximum probability of getting information about all the plaintext contained in the expression M. Finally, we define function pP_M^G, which employs the results obtained by applying the function $pGuess_M^G(T)$ in order to turn the expression M into a probabilistic pattern. We formally detail all these structures in the next subsections.

The novel equivalence relation captures when from two messages we can derive the same information and this information is obtained with the same probability in the case the adversary tries to cryptanalyze ciphered pieces of data. Formally, we verify that two expressions M and N are equivalent if they

[4] For the sake of simplicity, in every example we omit the knowledge from the parameters of p_{dec}.

yield the same probabilistic pattern (obtained through the functions pP^G_M and pP^G_N) and if the probabilities of getting information about the overall plaintext (expressed by the parameters $pMax^G_M$ and $pMax^G_N$) are equal. Throughout the paper we usually assume that set G expressing the initial knowledge of the adversary is empty. When we resort to such an assumption we omit G from the structures. Obviously, it is worth noting that as the adversary gets additional information, the enriched knowledge may be responsible for increasing the estimation given by function p_{dec}.

2.2 ***pKeys***

The first step of our procedure that aims at turning an expression M into a probabilistic pattern consists of generating set $\mathbf{pKeys}^G_M$. Such a set contains pairs $T._p$, where $T \subseteq \mathbf{Keys}$ is a set of keys syntactically occurring in M, and $p \in]0,1]$ is the probability of retrieving information useful to cryptanalyze ciphertexts obtained with the keys contained in T. The set $\mathbf{pKeys}^G_M$ is generated by the following two-step algorithm:

$$\begin{aligned}
&\mathbf{pKeys}^G_M = \{initKeys((M,G))._1\};\\
&addKeys((M,G),1);
\end{aligned}$$

where $initKeys : \mathbf{Exp} \rightarrow \mathcal{P}(\mathbf{Keys})$ takes as input an expression L and returns the set of keys recoverable from L through the entailment relation. Formally, we have $initKeys(L) = \{K \in \mathbf{Keys} \mid L \mapsto K\}$. Then, $addKeys(H,p)$, with $H \in \mathbf{Exp}$ and $p \in]0,1]$, is the following recursive procedure:

$$\begin{aligned}
&addKeys(H,p) ::= \\
&\quad \forall\, \{N\}_K \;:\; (H \mapsto \{N\}_K \;\wedge\; H \not\mapsto K) \text{ do begin}\\
&\qquad p' \qquad\qquad = p \cdot p_{dec}(\{N\}_K, H)\\
&\qquad L \qquad\qquad = (H,K)\\
&\qquad T \qquad\qquad = \{K \in \mathbf{Keys} \mid L \mapsto K\}\\
&\qquad \mathbf{pKeys}^G_M \quad = \mathbf{pKeys}^G_M \cup \{T._{p'}\}\\
&\qquad addKeys(L,p')\\
&\quad \text{end}
\end{aligned}$$

Initially, $\mathbf{pKeys}^G_M$ is set $\{initKeys((M,G))._1\}$, where $initKeys((M,G))$ is the set of keys that can be derived from M and G with probability 1 (i.e., the keys an adversary infers from the expression M and from the initial knowledge G without cryptanalysis attempts). In particular, $initKeys((M,G))$ contains each key K such that $(M,G) \mapsto K$.

Then, at each step, we add to $\mathbf{pKeys}_M^G$ sets of keys (obtained from expression M) that are somehow cracked. In particular, for each cryptanalysis strategy that an adversary may follow, $\mathbf{pKeys}_M^G$ contains the set of keys violated by following that strategy and the probability of cracking such keys. The procedure *addKeys* recursively adds to set $\mathbf{pKeys}_M^G$ the results of each cryptanalysis strategy.

Note that function $p_{dec}(\{N\}_K, H)$ (with $N \in \mathbf{Exp}, K \in \mathbf{Keys}$) computes the probability, for an adversary with knowledge H, of obtaining useful information from N without knowing the key K. In particular, H contains the initial knowledge G of the adversary, and the knowledge obtained by cryptanalyzing expression M.

Example 2.1 Given $M = ((\{\{(m, K)\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K), K')$, $\mathbf{pKeys}_M$ is initialized with the keys inferred from (M, G) with probability 1 through the entailment relation. Besides, we assume that the adversary has no initial knowledge, so we just have $M \mapsto K'$, and we start with $\mathbf{pKeys}_M = \{\{K'\}_{\cdot 1}\}$. Then, the *addKeys* procedure evaluates all the possible cryptanalysis sequences an adversary may follow, by adding at each step new elements to set $\mathbf{pKeys}_M$ (see Fig. 1). We observe that the set of keys $\{K', K, K_1, K_2\}$ appears three times in $\mathbf{pKeys}_M$ (see Fig. 2) with different probabilities. This is due to the alternative strategies the adversary may follow to obtain the plaintext.

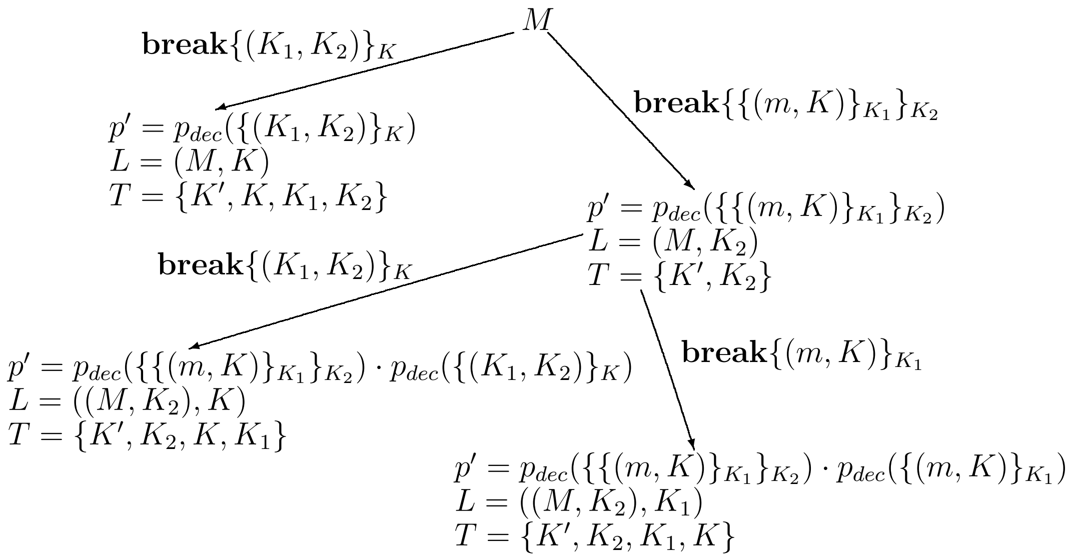

Fig. 1. Computation paths of $addKeys(M, 1)$

2.3 pGuess

Function $pGuess_M^G(T)$ computes the maximum probability for the adversary of cracking all the keys in T according to the best cryptanalysis strategy that can be followed to attack the expression M. Let $allKeys(M)$ be the set of

$$\mathbf{pKeys}_M = \left\{ \begin{array}{l} \{K'\}_{\cdot 1}, \\ \{K', K, K_1, K_2\}_{\cdot p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_2\}_{\cdot p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2})}, \\ \{K', K_2, K, K_1\}_{\cdot p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_2, K_1, K\}_{\cdot p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(m,K)\}_{K_1})} \end{array} \right\}$$

Fig. 2. $\mathbf{pKeys}_M$

all key symbols that occur in M; we define $\mathcal{D}_{pGuess^G_M} = \{T \subseteq \mathbf{Keys} \mid T \subseteq allkeys(M)\}$.

Then, function $pGuess^G_M : \mathcal{D}_{pGuess^G_M} \rightarrow]0,1]$ is formally defined as:

$$pGuess^G_M(T) = \max\{p \mid J_{\cdot p} \in \mathbf{pKeys}^G_M \ \wedge \ T \subseteq J\}.$$

It is worth noting that $pGuess^G_M(\emptyset) = 1$. Indeed, $\forall M \in \mathbf{Exp}$, we have $\emptyset \subseteq initKeys((M,G))$ and $initKeys((M,G))_{\cdot 1} \in \mathbf{pKeys}^G_M$.

Example 2.2 Consider again $M = ((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. We have that (see set $\mathbf{pKeys}_M$ in Fig. 2):

$$\begin{array}{ll} pGuess_M(\{K'\}) & = 1 \\ pGuess_M(\{K_2\}) & = \max \left\{ \begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(m,K)\}_{K_1}) \end{array} \right\} \\ & = \max \left\{ \begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \end{array} \right\} \\ pGuess_M(\{K', K, K_1, K_2\}) & = \max \left\{ \begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2} \cdot p_{dec}(\{(m,K)\}_{K_1}) \end{array} \right\} \end{array}$$

2.4 pMax

Given an expression M, parameter $pMax^G_M$ expresses the probability of getting the maximum information from M. Therefore, $pMax^G_M$ represents the maximum probability of guessing all the keys used in M. Formally, $pMax^G_M$ is derived in the following way:

$$pMax^G_M = \max\{p \mid J_{\cdot p} \in \mathbf{pKeys}^G_M \ \wedge \ allKeys(M) \subseteq J\}.$$

Given that $pMax_M^G$ represents the probability of guessing all the keys used in M, and by considering the definitions of $pMax_M^G$ and $pGuess_M^G$, we observe that $pMax_M^G$ can also be defined as:

$$pMax_M^G = pGuess_M^G(allKeys(M)).$$

Example 2.3 Consider $M = ((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. Since we have $allKeys(M) = \{K', K, K_1, K_2\}$, it follows that (see the values of $pGuess_M$ in Example 2.2):

$$\begin{aligned} pMax_M &= pGuess_M(\{K', K, K_1, K_2\}) \\ &= \max \left\{ \begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(m,K)\}_{K_1}) \end{array} \right\} \end{aligned}$$

2.5 pP

The family of functions pP turns expressions into probabilistic patterns. In particular, given an expression N contained in M, pP_M^G saves in a set T the set of keys needed to decrypt each ciphered block occurring in N, extracts the information contained in N, and associates to such a plaintext the maximum probability of obtaining it through the best cryptanalysis strategy that can be applied to M. To this end, we employ function $pGuess_M^G$ to compute the probability of cracking the keys needed to decrypt each ciphertext occurring in N. Hence, a probabilistic pattern contains plaintext (which can be inferred with a certain probability) instead of ciphertext.

The function $pP_M^G : \mathbf{Exp} \times \mathcal{D}_{pGuess_M^G} \rightarrow \mathbf{pPat}$ is defined inductively as follows:

$$\begin{array}{lll} pP_M^G(K,T) & = K._{pGuess_M^G(T)} & (K \in \mathbf{Keys}) \\ pP_M^G(m,T) & = m._{pGuess_M^G(T)} & (m \in \mathbf{String}) \\ pP_M^G((N_1,N_2),T) & = (pP_M^G(N_1,T), pP_M^G(N_2,T))._{pGuess_M^G(T)} & \\ pP_M^G(\{N\}_K,T) & = pP_M^G(N,T') & (T' = T \cup \{K\}) \end{array}$$

The probabilistic pattern that can be obtained from an expression N through the best cryptanalysis strategy applied to M is $pP_M^G(N,\emptyset)$. In the following, given an expression M, we use the abbreviation pP_M^G (with no arguments) to stand for $pP_M^G(M,\emptyset)$.

Example 2.4 Consider again $M = ((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. We have that [5]:
$pP_M = pP_M(((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K'), \emptyset) =$

[5] When equal to 1, we omit parameter p in every probabilistic pattern $P.p$.

$(pP_M((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), \emptyset), K')$
where from
$pP_M((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), \emptyset)$
we get (given $\bar{p} = pGuess_M(\{K_2, K_1\})$)
$pP_M(\{\{(m,K)\}_{K_1}\}_{K_2}, \emptyset) = (m_{\cdot\bar{p}}, K_{\cdot\bar{p}})_{\cdot\bar{p}}$
and (given $\hat{p} = pGuess_M(\{K\})$)
$pP_M(\{(K_1,K_2)\}_K, \emptyset) = (K_{1\cdot\hat{p}}, K_{2\cdot\hat{p}})_{\cdot\hat{p}}$.

Example 2.5 As another example, consider two expressions that yield the same probabilistic patterns and have two different $pMax$ values:

$$M = (\{m\}_K, \{n\}_K) \quad N = (\{m\}_K, \{n\}_{K'}), \quad K \neq K'.$$

We have that:

$$pP_M = (m_{\cdot\hat{p}}, n_{\cdot\hat{p}}),$$

where $\hat{p} = pGuess_M(\{K\}) = \max\{p_{dec}(\{m\}_K), p_{dec}(\{n\}_K)\}$. The intuition is that an adversary can get information contained in M by guessing K, which is used to cipher both blocks. On the other hand, if $pGuess_M(\{K\}) = pGuess_N(\{K\}) = pGuess_N(\{K'\})$ we also have that:

$$pP_N = (m_{\cdot\hat{p}}, n_{\cdot\hat{p}}) = pP_M.$$

Hence, M and N have the same probabilistic patterns, even if to get in clear the whole expression N an adversary should guess two different keys, namely K and K'. Such a difference is captured by the fact that:

$$pMax_M = pGuess_M(\{K\}) = \hat{p} \neq \hat{p}^2 = pGuess_N(\{K, K'\}) = pMax_N.$$

Therefore, $pMax$ is needed to express the overall probability of getting the entire expression in clear, while the probabilistic pattern is used to associate to each piece of information contained in an expression the probability to get it in clear.

2.6 Equivalence

Given the expressions M and N, we say that M and N are *probabilistically equivalent* ($M \approx N$) if they yield the same probabilistic pattern and if $pMax_M$ and $pMax_N$ are equal. Formally we have:

$$M \approx N \qquad \Leftrightarrow \qquad pP_M = pP_N \ \wedge \ pMax_M = pMax_N.$$

Intuitively, two expressions are probabilistically equivalent if one can derive from them the same information and this information is obtained with the same probability in the case it is ciphered with unknown keys.

Example 2.6 Consider $M = ((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$ and $N = ((\{(m,K)\}_{K_1}, \{(K_1,K_2)\}_K), K')$. On the one hand, we have that:

$$\mathbf{pKeys}_N = \left\{ \begin{array}{l} \{K'\}._1, \\ \{K', K, K_1, K_2\}._{p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_1, K, K_2\}._{p_{dec}(\{(m,K)\}_{K_1})} \end{array} \right\}$$

so if $p_{dec}(\{(m,K)\}_{K_1}) \leq p_{dec}(\{(K_1,K_2)\}_K)$ we have that $pGuess_N(\{K_1\}) = pGuess_N(\{K\}) = p_{dec}(\{(K_1,K_2)\}_K)$ and, given $\hat{p} = p_{dec}(\{(K_1,K_2)\}_K)$, we have $pP_N = (((m._{\hat{p}}, K._{\hat{p}})._{\hat{p}}, (K_1._{\hat{p}}, K_2._{\hat{p}})._{\hat{p}}), K')$.

On the other hand, from the previous examples and from the condition $p_{dec}(\{(m,K)\}_{K_1}) \leq p_{dec}(\{(K_1,K_2)\}_K)$ we obtain the probabilistic pattern $pP_M = (((m._{\hat{p}}, K._{\hat{p}})._{\hat{p}}, (K_1._{\hat{p}}, K_2._{\hat{p}})._{\hat{p}}), K')$ and, since $pMax_M = pMax_N = \hat{p}$, we also obtain $M \approx N$. In conclusion, we observe that ciphering the first block (m, K) of M with both keys K_1 and K_2 is not meaningful, since M is probabilistically equivalent to an expression where this information is ciphered with one of those keys only. In fact, an adversary could gain information about m by cryptanalyzing the second block $\{(K_1,K_2)\}_K$.

Example 2.7 Consider an adversary with initial knowledge $G = K_1$ and again the expressions:

$$M = ((\{\{(m,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$$
$$N = ((\{(m,K)\}_{K_1}, \{(K_1,K_2)\}_K), K').$$

We note that $M \not\approx N$, since the adversary gets in clear the entire expression N with probability 1. In fact, through the key K_1 the adversary can get all the other keys by means of the entailment relation. In particular, we have that:

$$\mathbf{pKeys}_N^G = \{\{K', K_1, K, K_2\}._1\}$$
$$\mathbf{pKeys}_M^G = \left\{ \begin{array}{l} \{K', K_1\}._1, \\ \{K', K_1, K_2, K\}._{p_{dec}(\{\{(m,K)\}_{K_1}\}_{K_2})}, \\ \{K', K_1, K, K_2\}._{p_{dec}(\{(K_1,K_2)\}_K)} \end{array} \right\}$$

Example 2.8 Consider expressions M and N of Example 2.5. As we have seen, M and N are not probabilistically equivalent, since they are associated with different $pMax$ values. Now, let us add to both expressions the ciphertext $\{o\}_{K'}$, thus getting the expressions $M' = ((\{m\}_K, \{n\}_K), \{o\}_{K'})$ and $N' = ((\{m\}_K, \{n\}_{K'}), \{o\}_{K'})$. In this case, given $pGuess_{M'}(\{K\}) = pGuess_{M'}(\{K'\}) = pGuess_{N'}(\{K\}) = pGuess_{N'}(\{K'\}) = \hat{p}$, we obtain that $pP_{M'} = pP_{N'} = ((m._{\hat{p}}, n._{\hat{p}})._{\hat{p}}, o._{\hat{p}})$ and $pMax_{M'} = pMax_{N'} = \hat{p}^2$, so that $M' \approx N'$.

This example shows that our equivalence relation is not conservative under the kind of operations as above. This is a reasonable consequence of the intuition behind the notion of equivalence among cryptographic expressions. Note that the information carried by expressions M' and N' is rather different from that expressed by expressions M and N. In order to get the plaintext from M' and N', an attacker has to break two different keys, similarly as seen in the case of expression N. Instead, for expression M it is sufficient to break key K only. Since the knowledge of the adversary changes by adding pieces of information to an expression, the probabilistic equivalence cannot be preserved by constructing (destructing) cryptographic expressions.

3 Indistinguishability through Probabilistic Similarity

The notion of probabilistic equivalence given above is very restrictive. In practice, it could be very difficult to find blocks that can be decrypted exactly with the same probability. As a consequence, two expressions containing the same information and yielding probabilistic patterns with similar probability (but not exactly the same) would not be probabilistically equivalent. Similarly, two different probabilistic patterns that can be deduced with negligible probabilities cannot be probabilistically equivalent.

In this section, we introduce a compatibility relation, called $\varepsilon-$*probabilistic similarity* ($\approx_\varepsilon$), which (i) approximates the probabilistic equivalence by introducing a tolerance to small differences of the probabilistic parameters associated to the probabilistic patterns, and (ii) allows for considering indistinguishable those ciphertexts that can be decrypted with a negligible probability.

We say that expressions M and N are $\varepsilon-$probabilistically similar ($M \approx_\varepsilon N$) if $pMax_M$ and $pMax_N$ are "almost the same" and if M and N are compatible, according to the notion of compatibility specified below. Formally, we have:

$$M \approx_\varepsilon N \qquad \Leftrightarrow \qquad pP_M \sim_\varepsilon pP_N \ \wedge \ |pMax_M - pMax_N| \leq \varepsilon.$$

The compatibility relation $\sim_\varepsilon$ for probabilistic patterns expresses when two probabilistic patterns are indistinguishable. Formally, it is defined as follows:

$$\boxed{\begin{array}{lll}
P_{\cdot p} \sim_\varepsilon Q_{\cdot p'} & if\ p, p' \leq \varepsilon & P_{\cdot p}, Q_{\cdot p'} \in \mathbf{pPat} \\
K_{\cdot p} \sim_\varepsilon K_{\cdot p'} & if\ |p - p'| \leq \varepsilon & K \in \mathbf{Keys} \\
m_{\cdot p} \sim_\varepsilon m_{\cdot p'} & if\ |p - p'| \leq \varepsilon & m \in \mathbf{String} \\
(P_{\cdot p_1}, Q_{\cdot p_2})_{\cdot p_3} \sim_\varepsilon (P'_{\cdot p'_1}, Q'_{\cdot p'_2})_{\cdot p'_3} & if\ |p_3 - p'_3| \leq \varepsilon & \wedge \\
 & \multicolumn{2}{r}{P_{\cdot p_1} \sim_\varepsilon P'_{\cdot p'_1} \ \wedge \ Q_{\cdot p_2} \sim_\varepsilon Q'_{\cdot p'_2}} \\
 & \multicolumn{2}{r}{P_{\cdot p_1}, Q_{\cdot p_2}, P'_{\cdot p'_1}, Q'_{\cdot p'_2} \in \mathbf{pPat}}
\end{array}}$$

According to such rules, note that two different pieces of information turn out to be indistinguishable if they are associated with probabilistic parameters that are smaller than the given tolerance ε. This is because the probability of revealing the difference between them is negligible. Informally, we can use this notion in order to consider as undecryptable a ciphertext that can be decrypted with a probability smaller then the fixed threshold. In practice, if the cryptosystem is secure enough (according to the security degree specified by the given tolerance ε), then each ciphertext is really a black box.

Example 3.1 Consider the expressions $M = \{m\}_K$ and $N = \{n\}_{K'}$. Given the associated probabilistic patterns $pP_M = m._{p_1}$ and $pP_N = n._{p_2}$ and a fixed threshold ε, we say that M and N are indistinguishable if $p_1, p_2 \leq \varepsilon$. In such a case we have that $pP_M \sim_\varepsilon pP_N$ and, since $pMax_M = p_1 \leq \varepsilon$ and $pMax_N = p_2 \leq \varepsilon$, we have that $|pMax_M - pMax_N| \leq \varepsilon$. Therefore, it follows $M \approx_\varepsilon N$.

Example 3.2 Consider $M = \{m\}_K$ and $N = \{m\}_{K'}$. If we assume $p_1 = p_{dec}(\{m\}_K)$ and $p_2 = p_{dec}(\{m\}_{K'})$, we have that $pP_M = m._{p_1}$, $pP_N = m._{p_2}$, $pMax_M = p_1$ and $pMax_N = p_2$. If p_1 and p_2 are similar, but not exactly the same, then given a tolerance ε such that $|p_1 - p_2| \leq \varepsilon$, we have that $M \approx_\varepsilon N$ but $M \not\approx N$.

Example 3.3 Consider $M = (\{m\}_K, \{K\}_{K'})$ and $N = (\{m\}_K, K)$. If we consider K' as a weak key, we may have that $p_{dec}(\{m\}_K) < p_{dec}(\{K\}_{K'}) = p$ and, therefore, $pP_M = (m._p, K._p)$. Moreover, we also have $pP_N = (m, K)$. If the probability of violating K' is close to 1, and, in particular, if $(1 - p) \leq \varepsilon$, we have that $M \approx_\varepsilon N$. Intuitively, M is a ciphered variant of N that can be easily cracked.

The following proposition holds.

Proposition 3.4 *Given* $M, N \in \mathbf{Exp}$ *it holds that:*

$$M \approx N \qquad \Rightarrow \qquad M \approx_\varepsilon N \quad \forall \varepsilon \in [0,1].$$

Proof. *By definition of* $\sim_\varepsilon$ *it follows that* $pP_M = pP_N \;\Rightarrow pP_M \sim_\varepsilon pP_N \quad \forall \varepsilon \in [0,1]$. *Finally, we also have that* $pMax_M = pMax_N \;\Rightarrow\; |pMax_M - pMax_N| \leq \varepsilon \quad \forall \varepsilon \in [0,1]$. ■

By the definition of $\sim_\varepsilon$ the following proposition also holds.

Proposition 3.5 *Given* $M, N \in \mathbf{Exp}$ *it holds that:*

$$M \approx N \qquad \Leftrightarrow \qquad M \approx_0 N.$$

4 Perfect Cryptography vs. Ideal Encryption

In this section we show how our notion of similarity is related to a classical Dolev-Yao equivalence relation defined in an environment where perfect cryptography is assumed. In particular, given a notion of ideal encryption, we will show that if two expressions are equivalent within a classical Dolev-Yao model that relies on perfect cryptography, then the two expressions will also be probabilistically similar if we assume ideal encryption.

4.1 Equivalence within Perfect Cryptography

In [1], Abadi and Rogaway define an equivalence relation for cryptographic expressions within a formal model where perfect cryptography is assumed. The set **Pat** of *patterns* is defined as an extension of the set of expressions that employs the new symbol $\otimes$ representing a ciphertext that an adversary cannot decrypt.

$P, Q ::=$	patterns
K	key (for $K \in \mathbf{Keys}$)
m	string (for $m \in \mathbf{String}$)
(P, Q)	pair
$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
$\otimes$	undecryptable text

Intuitively, a pattern is an expression that may contain some parts that an adversary cannot decrypt. They define a function p that, given a set of keys T and an expression M, computes the pattern that an adversary can obtain from M if the initial knowledge is the set of keys T.

$$
\begin{array}{lll}
p(K, T) & = K & (\text{for } K \in \mathbf{Keys}) \\
p(m, T) & = m & (\text{for } m \in \mathbf{String}) \\
p((M, N), T) & = (p(M, T), p(N, T)) & \\
p(\{M\}_K, T) & = \begin{cases} \{p(M, T)\}_K & \text{if } K \in T \\ \otimes & \text{otherwise} \end{cases} &
\end{array}
$$

Moreover, they define $pat(M)$, which expresses the pattern obtained from an expression M without knowing a priori any auxiliary set T of keys. Formally, $pat(M) = p(M, initKeys(M))$. For example, $pat((\{\{K_1\}_{K_2}\}_{K_3}, K_3)) = (\{\otimes\}_{K_3}, K_3)$.

Finally, by abstracting from the initial knowledge, they say that two expressions are *equivalent* if they yield the same pattern:

$$M \cong N \qquad \Leftrightarrow \qquad pat(M) = pat(N).$$

For example, we have $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \cong (\{\{m\}_{K_1}\}_{K_3}, K_3)$ since both expressions yield the pattern $(\{\otimes\}_{K_3}, K_3)$.

In [20] we defined a probabilistic equivalence relation that is conservative with respect to the one introduced by Abadi and Rogaway. In this paper we decided to define a slight variant of that relation, which we consider more intuitive: two expressions should be equivalent when they yield the same information, and when such information can be extracted with the same (or similar) probabilities. By taking such a choice we lost the conservativeness with the equivalence of [1]. For example, if we take the two expressions $(\{m\}_K, K)$ and (m, K), it turns out that $(\{m\}_K, K) \not\cong (m, K)$, since the two expressions yield different patterns. However, the two expressions contain the same information, which is extractable with the same probability (in this case with probability 1), so we have that $(\{m\}_K, K) \approx (m, K)$ (note that the probabilistic pattern associated with both expressions is $(m._1, K._1)$).

4.2 Ideal Encryption

The notion of *ideal encryption* intuitively assumes that it should be hard for the adversary to decrypt a message ciphered with an unknown key. In other words, the probability of breaking an encrypted message that cannot be derived in the classical Dolev-Yao model should be *small*. We formalize the concept of small probabilities by introducing the definition of *negligible* function (see, e.g., [13]).

Definition 4.1 A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible*, if for any polynomial q, $\exists \eta_0 \in \mathbb{N} : f(\eta) \leq \frac{1}{q(\eta)} \quad \forall \eta > \eta_0$.

Then, the *ideal encryption* hypothesis assumes that p_{dec} must be a negligible function.

Definition 4.2 An encryption scheme is ideal if and only if

$$\begin{aligned}
&\forall \mathcal{A}, \quad \forall \{N\}_K \in \mathbf{Exp}, \quad \forall G \in \mathbf{Exp} : G \not\mapsto K, \quad \forall \text{ polynomial } q \\
&\exists \eta_0 \in \mathbb{N} : \\
&p_{dec}(\{N\}_K, G) \leq \tfrac{1}{q(\eta)} \quad \forall \eta > \eta_0.
\end{aligned}$$

As a consequence, if the assumption of ideal encryption holds, given $\{N\}_K, G \in$ **Exp** such that $G \not\mapsto K$, we also have that for all $\mathcal{A}$:

$$Pr[m \leftarrow \mathcal{A}(\{m\}_K, G)] \leq \frac{1}{q(\eta)} \quad \forall \eta > \eta_0.$$

By following an approach also used in [7], we show that a result holding in the perfect cryptography scenario also holds in our model if the ideal encryption assumption is taken.

Theorem 4.3 *Given $M, N \in$ **Exp**, if the assumption of ideal encryption holds for a natural η_0, taken a polynomial q and a natural parameter $\eta > \eta_0$, then:*

$$M \cong N \quad \Rightarrow \quad M \approx_\varepsilon N \quad \forall \varepsilon > \frac{1}{q(\eta)}.$$

Proof. *By the assumption of ideal encryption, either $pMax_M = pMax_N = 1$ or $pMax_M, pMax_N \leq \frac{1}{q(\eta)} < \varepsilon$. As a consequence, given $M \cong N$, in order to prove that $M \approx_\varepsilon N$ we just need to check that $pP_M \sim_\varepsilon pP_N$. The statement derives by structural induction on the expression M and by observing that, by hypothesis, $M \cong N \Rightarrow pat(M) = pat(N)$. In the following, we denote by T_M the set $initKeys(M)$ and by T_N the set $initKeys(N)$.*

(i) $pat(M) = pat(N) = K \qquad K \in$ **Keys**
$\Rightarrow$
$pP_M = pP_N = K._1 \Rightarrow pP_M \sim_\varepsilon pP_N \quad \forall \varepsilon \in [0,1]$.

(ii) $pat(M) = pat(N) = m \qquad m \in$ **String**
$\Rightarrow$
$pP_M = pP_N = m._1 \Rightarrow pP_M \sim_\varepsilon pP_N \quad \forall \varepsilon \in [0,1]$.

(iii) $pat(M) = pat(N) = \otimes$
$\Rightarrow$ *by the ideal encryption assumption*
$pP_M = Q._p,\ pP_N = Q'._{p'} \ \wedge\ p, p' \leq \varepsilon \Rightarrow pP_M \sim_\varepsilon pP_N$.

(iv) $pat(M) = \{p(L, T_M)\}_K = \{p(L', T_N)\}_K = pat(N)$
$\Rightarrow p(L, T_M) = p(L', T_N)$
$\Rightarrow$ *by induction hypothesis*
$pP_M(L, \emptyset) \sim_\varepsilon pP_N(L', \emptyset) \Rightarrow pP_M = pP_M(L, \emptyset) \sim_\varepsilon pP_N(L', \emptyset) = pP_N \Rightarrow pP_M \sim_\varepsilon pP_N$.

(v) $pat(M) = (p(L_1, T_M), p(L_2, T_M)) = (p(L'_1, T_N), p(L'_2, T_N)) = pat(N) \Rightarrow$
$p(L_1, T_M) = p(L'_1, T_N) \ \wedge\ p(L_2, T_M) = p(L'_2, T_N)$
$\Rightarrow$ *by induction hypothesis*
$pP_M(L_1, \emptyset) \sim_\varepsilon pP_N(L'_1, \emptyset) \ \wedge\ pP_M(L_2, \emptyset) \sim_\varepsilon pP_N(L'_2, \emptyset) \Rightarrow$
$pP_M = (pP_M(L_1, \emptyset), pP_M(L_2, \emptyset))._1 \sim_\varepsilon (pP_N(L'_1, \emptyset), pP_N(L'_2, \emptyset))._1 = pP_N$
$\Rightarrow pP_M \sim_\varepsilon pP_N$. ■

In general, the inverse implication of Theorem 4.3 does not hold. Consider for example the expressions $M = (\{m\}_K, K)$ and $N = (m, K)$. As we have seen, the two expressions are probabilistically equivalent, i.e. $M \approx N$, and, by Proposition 3.4, also probabilistically similar, i.e. $M \approx_\varepsilon N\ \forall \varepsilon \in [0,1]$. However, since the two expressions yield different patterns, they cannot be equivalent according to the Abadi-Rogaway equivalence relation. The result does not change even in the case of ideal encryption, since the probabilistic similarity holds for all $\varepsilon \in [0,1]$.

5 Related Work

The treatment of cryptographic operations within formal models is covered by a quite large body of literature, but most of these efforts do not consider cryptographic operations in an imperfect cryptography scenario.

This work represents a step toward the definition of a formal language with cryptographic primitives and conditional statements for analyzing both unwanted disclosure of data due to the nature of the protocols and information leakage due to the nature of the cryptographic means. In the literature, both probability and computational complexity are studied in formal settings.

Process algebra and computational view of cryptography are combined in [16] where, in the setting of a subset of asynchronous π-calculus, an asymptotic notion of probabilistic equivalence is defined. The observational equivalence defined in terms of such a notion can be related to polynomial time statistical tests, i.e. equivalent processes are indistinguishable from the viewpoint of polynomial time adversaries. Security is then stated in terms of indistinguishability between the protocol under analysis and an idealized protocol specification. More recently, a definition of probabilistic noninterference which includes a computational case has been defined in [4] in the setting of asynchronous probabilistic reactive systems. In particular, computational noninterference means that the advantage of the external observer (which interacts with the system under analysis) for a correct guess of the interfering adversary behavior is a negligible function. A formal notion of computational indistinguishability is also defined in [15] on the basis of a simple model where public outputs are observed in order to infer the content of secret inputs. Finally, [7] compares the classical Dolev-Yao adversary with an enhanced computational adversary which can guess the key for decrypting an intercepted message (albeit only with negligible probability). The two adversaries are shown to be equivalent with respect to a secrecy property.

We also point out that probabilistic notions of security as well as approximate security properties can be found in the literature (see, e.g., [12,9,3,2]), but they do not relate probability and cryptographic primitives.

6 Conclusions

In this paper we put the basis for defining a formal cryptographic language where (*i*) information leaks due to the weaknesses of the cryptographic primitives can be estimated by employing conditional statements and the equivalence relation presented in Sect. 2, and (*ii*) probabilistic covert channels can be revealed by verifying noninterference security properties (as done, e.g., in [9,3]). In particular, the approximate notion of indistinguishability of Sect. 3 can be used (together with an approximate definition of noninterference) to verify whether security properties of cryptographic protocols can be guaranteed at a reasonable degree.

We did not sift through the details of the estimation of the information leakage which is formally expressed by function p_{dec}. We have partially mitigated the effect of such an abstraction by relaxing the probabilistic equivalence through an approximate similarity relation, which allows us to relate expressions that can be considered equivalent up to a given tolerance ε. Moreover, similarly as the soundness result shown in [1], it turns out that if p_{dec} is a negligible function, i.e. the encryption scheme is ideal, equivalence in a formal setting stating perfect cryptography implies similarity in our framework.

References

[1] M. Abadi, P. Rogaway, *"Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)"*, in Proc. of 1st IFIP Int. Conf. on Theoretical Computer Science, Springer LNCS 1872:3-22, 2000.

[2] A. Aldini, M. Bravetti, A. Di Pierro, R. Gorrieri, C. Hankin, H. Wiklicky, *"Two Formal Approaches for Approximating Noninterference Properties"*, Foundations of Security Analysis and Design II, R. Focardi and R. Gorrieri, eds., Springer LNCS 2946:1-43, 2004.

[3] A. Aldini, M. Bravetti, R. Gorrieri, *"A Process-algebraic Approach for the Analysis of Probabilistic Non-interference"*, Journal of Computer Security, vol. 12(2), IOS Press, 2004.

[4] M. Backes, B. Pfitzmann, *"Computational Probabilistic Non-interference"*, in Proc. of 7th European Symposium on Research in Computer Security, Springer LNCS 2502:1-23, 2002.

[5] E. Biham, A. Biryukov, A. Shamir, *"Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials"*, in Advances in Cryptology, EUROCRYPT'99, Springer LNCS 1592:12-23, 1999.

[6] N. Borisov, I. Goldberg, D. Wagner, *"Intercepting Mobile Communications: The Insecurity of 802.11"*, in Proc. of 7th Int. Conf. on Mobile Computing and Networking, MOBICOM'01, ACM Press, pp. 180-189, 2001.

[7] P. Degano, R. Zunino, *"A Note on the Perfect Encryption Assumption in a Process Calculus"*, to appear in Foundations of Software Science and Computation Structures (FOSSACS'04), Springer Verlag, 2004.

[8] R. A. DeMillo, N. A. Lynch, M. Merritt, *"Cryptographic Protocols"*, in Proc. of the 14th Annual ACM Symposium on Theory of Computing, ACM Press, pp. 383-400, 1982.

[9] A. Di Pierro, C. Hankin, H. Wiklicky, *"Approximate Non-Interference"*, in Proc. of 15th Computer Security Foundations Workshop, IEEE CS Press, pp. 1-17, 2002.

[10] D. Dolev, A. Yao, *"On the Security of Public-key Protocols"*, IEEE Transactions on Information Theory 29:198-208, 1983.

[11] A. Durante, R. Focardi, R. Gorrieri, *"A Compiler for Analysing Cryptographic Protocols Using Non-Interference"*, ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 9(4):489-530, 2000.

[12] J. W. Gray III, *"Toward a Mathematical Foundation for Information Flow Security"*, Journal of Computer Security 1:255-294, 1992.

[13] J. Herzog, *"A Computational Interpretation of Dolev-Yao Adversaries"*, in Proc. of 3rd Int. Workshop on Issues in the Theory of Security (WITS'03), 2003.

[14] R. A. Kemmerer, *"Analyzing Encryption Protocols using Formal Verification Techniques"*, IEEE Journal on Selected Areas in Communications, 7(4):448-457, 1989.

[15] P. Laud, *"Semantics and Program Analysis of Computationally Secure Information Flow"*, in Proc. of 10th European Symposium on Programming (ESOP'01), Springer LNCS 2028:77-91, 2001.

[16] P. Lincoln, J. C. Mitchell, M. Mitchell, A. Scedrov, *"A Probabilistic Poly-Time Framework for Protocol Analysis"*, in Proc. of 5th ACM Conf. on Computer and Communications Security, ACM Press, pp. 112-121, 1998.

[17] J. K. Millen, S. C. Clark, S. B. Freedman, *"The Interrogator: Protocol Security Analysis"*, IEEE Transactions on Software Engineering, SE-13(2):274-288, 1987.

[18] L. C. Paulson, *"The Inductive Approach to Verifying Cryptographic Protocols"*, Journal of Computer Security, 6(1-2):85-128, 1998.

[19] S. Schneider, *"Security Properties and CSP"*, in IEEE Symposium on Security and Privacy, IEEE CS Press, pp. 174-187, 1996.

[20] A. Troina, A. Aldini, R. Gorrieri, *"A Probabilistic Formulation of Imperfect Cryptography"*, in Proc. of 1st Int. Workshop on Issues in Security and Petri Nets, WISP'03, 2003.