

# Towards a Formal Treatment of Secrecy against Computational Adversaries

Angelo Troina<sup>1</sup>, Alessandro Aldini<sup>2</sup>, and Roberto Gorrieri<sup>3</sup>

<sup>1</sup> Dipartimento di Informatica, University of Pisa,  
Via F. Buonarroti 2, 56127 - Pisa, Italy  
`troina@di.unipi.it`

<sup>2</sup> Istituto STI, University of Urbino,  
Piazza della Repubblica 13, 61029 - Urbino, Italy  
`aldini@sti.uniurb.it`

<sup>3</sup> Dipartimento di Scienze dell'Informazione, University of Bologna,  
Mura Anteo Zamboni 7, 40127 - Bologna, Italy  
`gorrieri@cs.unibo.it`

**Abstract.** Polynomial time adversaries based on a computational view of cryptography have additional capabilities that the classical Dolev-Yao adversary model does not include. To relate these two different models of cryptography, in this paper we enrich a formal model for cryptographic expressions, originally based on the Dolev-Yao assumptions, with computational aspects based on notions of probability and computational power. The obtained result is that if the cryptosystem is robust enough, then the two adversary models turn out to be equivalent. As an application of our approach, we show how to determine a secrecy property against the computational adversary.

## 1 Introduction

The recent literature concerning the analysis of security protocols reveals an increasing interest towards the compatibility problem between the computational approach, followed by the cryptography community, and the approach based on the Dolev-Yao model, which is instead followed by the formal methods community (see, e.g., [2, 5, 11, 22, 15, 14, 18, 6]). In particular, it has been widely recognized that a sort of computational view of cryptography must be introduced in the formal approaches to security analysis based on a purely formal treatment of cryptographic operations. The classical Dolev-Yao model, which is based on the perfect cryptography assumption and the restricted expressive power of the adversary [8], favours a convenient application of formal methods that treat cryptographic operations as purely formal. In this view, an encrypted message, which is a formal expression, can be suitably analyzed through techniques borrowed from the fields of, e.g., modal logic and process algebra [16, 12, 10, 19, 17, 9]. On the contrary, such a model does not take into account that the adversary has (limited) computational resources which can be exploited to obtain data in a way that cannot be captured by, e.g., standard inference rules. The adversary advantage is instead based on notions of probability and computational power.

On the basis of such considerations, we aim at relaxing the strict requirements of the Dolev-Yao approach to cryptography. In order to overcome the limitations of such requirements, we take into account the probability for a polynomial time adversary of attacking with success a message encrypted with a secret key. While in a Dolev-Yao setting such a possibility is simply disregarded – a message encrypted with an unknown key is a black box – in a real scenario an adversary with a suitable knowledge may have a good chance of obtaining useful information from a ciphertext that, from a purely formal standpoint, is considered to be a black box. By considering the probability of cryptanalyzing a ciphertext, we compare cryptographic expressions through a suitable notion of indistinguishability, which has been introduced in [21]. Such a notion, which is based on a similarity relation, states whether a polynomial time adversary with a certain initial knowledge has a non-negligible probability of distinguishing two different cryptographic expressions. As a simple example, expressions  $\{M\}_K$  and  $\{rubbish\}_K$  are almost the same if  $K$  is secret and the encryption scheme is *ideal* according to a computational view of what, in practice, perfect cryptography stands for (see, e.g., [2, 11]).

In this paper, we show that the definition of similarity for cryptographic expressions corresponds to the classical Dolev-Yao based notion of equivalence in the case a suitably defined encryption scheme is used that, intuitively, turns out to be robust against any cryptanalysis attack conducted by a polynomial-time adversary. In practice, the intuition is that if the cryptosystem is robust enough, then a computational adversary with a limited amount of resources has the same expressive power of an adversary that does not use cryptanalysis to obtain data. As an application, we show that our notion of similarity can be used to determine the secrecy degree of a message within an encrypted expression.

The rest of the paper is organized as follows. First, we describe how we extended the Dolev-Yao formal model with probabilistic information used to estimate the probability for a polynomial-time adversary of obtaining meaningful information from a ciphertext (Sect. 2). Then, we show a similarity relation that allows cryptographic expressions to be compared from the viewpoint of a polynomial-time adversary (Sect. 3). Afterwards, we present the main theorem showing that such a similarity relation corresponds to the equivalence relation of the Dolev-Yao model in the case the encryption scheme is robust enough (Sect. 4). As an example, we show an application of such an approach to a secrecy problem in system security analysis (Sect. 5). Finally, we conclude the paper by discussing some related work (Sect. 6) and future work (Sect. 7).

## 2 Equivalence and Computational Adversary

We base our formal model on the Dolev-Yao encryption model defined by Abadi and Rogaway [2]. In this setting, we formulate an extension of the classical equivalence relation among cryptographic expressions that allows for relating those expressions that 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 account cryptanalysis attacks.

## 2.1 Setting the Context

As a preliminary to our extension, we now introduce the machinery needed to compare cryptographic 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  $m, n, \dots$ ), **Keys** to denote a fixed, non-empty set of key symbols (ranged over by  $K, K', K'', \dots$  and  $K_1, K_2, K_3, \dots$ ), 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 $K \in \mathbf{Keys}$ )

Intuitively,  $(M, N)$  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  $(\{(m, K)\}_{K_1}, K_1)$ .

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:

$M \mapsto M$	
$M \mapsto N_1 \wedge M \mapsto N_2$	$\Rightarrow M \mapsto (N_1, N_2)$
$M \mapsto (N_1, N_2)$	$\Rightarrow M \mapsto N_1 \wedge M \mapsto N_2$
$M \mapsto N \wedge M \mapsto K$	$\Rightarrow M \mapsto \{N\}_K$
$M \mapsto \{N\}_K \wedge M \mapsto K$	$\Rightarrow M \mapsto N$

In essence,  $M \mapsto N$  expresses what the adversary obtains from  $M$  without any prior knowledge of its content. 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  $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \not\mapsto K_1$ . The entailment relation models the expressive power of the adversary based on the Dolev-Yao model and includes all the operations that such an adversary can execute to construct ciphertexts or extract plaintexts.

Our extension consists in taking into account the possibility for an adversary of obtaining meaningful information from a ciphertext  $\{M\}_K$  without knowing the key  $K$ . To this purpose, we introduce the *probabilistic pattern*  $P_p$ , which represents an expression  $P$  that does not contain undecryptable blocks and is associated with a parameter  $p \in ]0, 1]$ , which 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 $K \in \mathbf{Keys}$ )
$m.p$	string (for $m \in \mathbf{String}$ )
$(P.p, Q.p).p$	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 a computational polynomial time adversary  $\mathcal{A}$  and an initial knowledge  $G$ , the probabilistic pattern associated with expression  $\{m\}_K$  is expressed in terms of the probability of obtaining information about  $m$ , denoted by  $m.p_{dec}(\{m\}_K, G)$ . Function  $p_{dec}(\{m\}_K, G)$  returns the probability of obtaining meaningful information from the ciphertext  $\{m\}_K$  by exploiting the initial knowledge  $G$ . More formally, an adversary  $\mathcal{A}$  with polynomially timed resources and knowledge  $G$  has a probability  $Pr$  at most equal to the value expressed by  $p_{dec}$  of retrieving  $K$  from  $\{m\}_K$ :

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

Note that the outcome of  $p_{dec}$  is a value strictly greater than 0, because, even if with small probability, an adversary may 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 (probabilistic) algorithm, executing in polynomial time, that makes computations on ciphered blocks in order to get information about the ciphering key and the contained plaintext (see, e.g., [11]). Note that the classical Dolev-Yao adversary obtains  $K$  from  $\{m\}_K$  if and only if  $K$  can be derived from  $G$ : If  $G \mapsto K$ , then  $p_{dec}(\{m\}_K, G) = 1$ . On the other hand, in a computational model assuming ideal encryption [11] or type-0 secure encryption scheme [2],  $p_{dec}$  is a negligible function, as it turns out that the probability of guessing information that cannot be derived through the 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 it holds that the expressive power of the computational adversary is limited by that of the Dolev-Yao adversary and vice versa.

The outcome of function  $p_{dec}$  represents the starting point for the computation of the probability of cracking a ciphered block. Consider, e.g., expression  $(\{\{m\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K)$ . What is the probability of getting information about  $m$  in the case no prior knowledge is available? An immediate answer could be  $p_{dec}(\{\{m\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{m\}_{K_1})$ <sup>4</sup>, 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

<sup>4</sup> For the sake of simplicity, we omit the knowledge  $G$  whenever either  $G$  is equal to the empty set or the content of  $G$  is clear from the context.

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 adopted by the adversary. By considering the worst case, we always associate to a ciphered block the maximum probability of getting information about it, i.e. we take into account the best cryptanalysis strategy from the adversary standpoint. To this end, we analyze all the possible cryptanalysis paths that the adversary can follow. In the next section, we describe through an illustrative example the structures and the functions used to determine the best cryptanalysis strategy [21].

## 2.2 Cryptanalyzing a ciphertext

The methodology that aims at turning an expression  $N$  into a probabilistic pattern  $N.p$  consists of four steps. In this section we illustrate each step of the methodology through an illustrative example. We consider the expression  $N = (\{\{m\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K)$  and we assume that the initial knowledge  $G$  does not allow the adversary to derive any information from  $N$  through the entailment relation.

The first step of our methodology consists in computing the keys that can be obtained from the expression, possibly with (without) cryptanalysis attempts. In our example, by hypothesis it is not possible to obtain keys through the entailment relation. In other words, the expression is a black box from the viewpoint of a classical Dolev-Yao adversary. In fact, we have that  $K$  can be derived with a probability based on  $p_{dec}(\{(K_1, K_2)\}_K)$ . Through such a single cryptanalysis, the adversary obtains  $K_1$  and  $K_2$  too. Alternatively, the adversary may try to obtain  $K_2$  by attacking  $\{\{m\}_{K_1}\}_{K_2}$  with a success probability equal to  $p_{dec}(\{\{m\}_{K_1}\}_{K_2})$ . Afterwards,  $\{m\}_{K_1}$  may be cracked in a similar way, and so on. In essence, several different strategies can be adopted to derive the keys contained in  $N - K, K_1$ , and  $K_2$  – from  $N$  itself, and each of such strategies must be evaluated.

Formally, given an expression  $M$  and the initial knowledge  $G$ , we denote by  $\mathbf{pKeys}_M^G$  a set of pairs of the form  $T.p$ , where  $T \subseteq \mathbf{Keys}$  is a set of keys syntactically occurring in  $M$ , and  $p \in ]0, 1]$  is the probability of retrieving the keys contained in  $T$  through a certain strategy. Set  $\mathbf{pKeys}_M^G$  is generated through the following two-phase algorithm:

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

In the first phase,  $\mathbf{pKeys}_M^G$  is initialized with the set of keys that can be derived from  $M$  and  $G$  without cryptanalysis attempts. Formally,  $initKeys : \mathbf{Exp} \rightarrow \mathcal{P}(\mathbf{Keys})$  takes an expression  $L$  and returns the set of keys recoverable from  $L$  through the entailment relation. Hence,  $initKeys(L) = \{K \in \mathbf{Keys} \mid L \mapsto K\}$ . Then, in the second phase, the probabilities of retrieving the remaining keys contained in  $M$  are calculated. Formally,  $addKeys(H, p)$ , with  $H \in \mathbf{Exp}$  and

$p \in ]0, 1]$ , is defined through the following algorithm:

```

addKeys(H, p) ::=
  ∀ {N}ₖ : (H ↦ {N}ₖ ∧ H ↯ K) do begin
    p'           = p · pdec({N}ₖ, H)
    L           = (H, K)
    T           = {K ∈ Keys | L ↦ K}
    pKeysMG    = pKeysMG ∪ {T, p'}
    addKeys(L, p')
  end

```

At each step of the algorithm above, a cryptanalysis is performed that reveals, with a certain probability, new keys obtained from  $M$ . In particular, for each cryptanalysis strategy,  $\mathbf{pKeys}_M^G$  contains the set of keys violated by following that strategy and the probability of cracking such keys.

The second step of our methodology consists in computing the maximum probability of retrieving from an expression a given set of keys by following the best cryptanalysis strategy. In our example, the maximum probability of guessing  $K$  is equal to the probability of cracking  $\{(K_1, K_2)\}_K$ , because this is the only strategy that can be followed to obtain  $K$ . On the other hand, the maximum probability of guessing  $K_2$  is the maximum between  $p_{dec}(\{\{m\}_{K_1}\}_{K_2})$  and  $p_{dec}(\{(K_1, K_2)\}_K)$ , which are the probabilities associated with the two possible strategies that can be followed to obtain  $K_2$ .

Formally, given an expression  $M$ , the initial knowledge  $G$ , and a set  $T$  of keys included in  $M$ , we denote by  $pGuess_M^G(T)$  the maximum probability of cracking all the keys in  $T$  according to the best cryptanalysis strategy that can be followed to attack  $M$ . Denoted  $Keys(M)$  the set of keys occurring in  $M$  and given the set  $\mathcal{D}_{pGuess_M^G} = \{T \subseteq \mathbf{Keys} \mid T \subseteq Keys(M)\}$ , we define  $pGuess_M^G : \mathcal{D}_{pGuess_M^G} \rightarrow ]0, 1]$  as:

$$pGuess_M^G(T) = \max\{p \mid J_p \in \mathbf{pKeys}_M^G \wedge T \subseteq J\}.$$

Note that  $pGuess_M^G(\emptyset) = 1$  since  $initKeys((M, G))_1 \in \mathbf{pKeys}_M^G$  and  $\emptyset \subseteq initKeys((M, G))$ .

The third step of our methodology consists in computing the maximum probability of retrieving all the information contained in a ciphertext. In our example, that means we want to evaluate the maximum probability of getting  $m$ ,  $K$ ,  $K_1$ , and  $K_2$ . As it is easy to see, such a probability is equal to  $p_{dec}(\{(K_1, K_2)\}_K)$ , because through such a single cryptanalysis it is possible to obtain all the keys used within  $N$ .

Formally, given an expression  $M$  and the initial knowledge  $G$ , we denote by  $pMax_M^G$  the maximum probability of getting the information contained in  $M$ :

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

The fourth step of our methodology consists in turning each ciphered block of an expression into a probabilistic pattern. The obtained probabilistic patterns

associate each plaintext with the maximum probability of obtaining it. In our example, the ciphertext  $\{(K_1, K_2)\}_K$  is turned into the probabilistic pattern  $(K_{1..p}, K_{2..p})_{.p}$  where  $p = pGuess_M^G(\{K\}) = p_{dec}(\{(K_1, K_2)\}_K)$  and the ciphertext  $\{\{m\}_{K_1}\}_{K_2}$  is turned into the probabilistic pattern  $m_{.pGuess_M^G(\{K_1, K_2\})}$ .

Formally, given an expression  $M$ , the initial knowledge  $G$ , and an expression  $M'$  contained in  $M$ , we denote by  $pP_M^G(M', T)$  a function that (i) returns in  $T$  the set of keys needed to obtain the plaintext contained in  $M'$  and (ii) associates to such a plaintext the maximum probability of obtaining it through the best cryptanalysis strategy that can be applied to  $M$ . Function  $pP_M^G : \mathbf{Exp} \times \mathcal{D}_{pGuess_M^G} \rightarrow \mathbf{pPat}$  is defined inductively as follows:

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

Finally, given an expression  $M$  and the initial knowledge  $G$ , the probabilistic pattern associated to  $M$  is given by  $pP_M^G(M, \emptyset)$ . In the following, we use the abbreviation  $pP_M^G$  (with no arguments) to stand for  $pP_M^G(M, \emptyset)$ .

The values  $pMax_M^G$  and  $pP_M^G$  yield different information that are both meaningful to relate cryptographic expressions. Consider the following expressions:

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

which yield the same probabilistic patterns. Indeed <sup>5</sup>:

$$pP_M = (m_{.\hat{p}}, n_{.\hat{p}})_{.1},$$

where  $\hat{p} = pGuess_M(\{K\}) = \max\{p_{dec}(\{m\}_K), p_{dec}(\{n\}_{K'})\}$ . The intuition is that an adversary can crack  $M$  by guessing  $K$ , which is used to cipher both blocks. However, if  $pGuess_M(\{K\}) = pGuess_N(\{K\}) = pGuess_N(\{K'\})$  we also have that:

$$pP_N = (m_{.\hat{p}}, n_{.\hat{p}})_{.1}.$$

Hence,  $M$  and  $N$  have the same probabilistic pattern, 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,  $pMax_M^G$  is needed to express the overall probability of getting the whole plaintext, while  $pP_M^G$  is needed to associate each piece of information contained in an expression with the probability of getting it in clear.

In the following, we show how the information computed through the methodology surveyed above can be exploited to compare cryptographic expressions.

<sup>5</sup> We assume an empty knowledge and omit  $G$ .

### 2.3 Probabilistic Equivalence

Given the expressions  $M$  and  $N$  and an initial knowledge  $G$ , we say that  $M$  and  $N$  are *probabilistically equivalent* ( $M \approx^G N$ ) if they yield the same probabilistic pattern and if  $pMax_M^G$  and  $pMax_N^G$  are equal.

**Definition 1.**  $M \approx^G N \Leftrightarrow pP_M^G = pP_N^G \wedge pMax_M^G = pMax_N^G$ .

*Example 1.* Consider the expressions  $N = (\{m\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K)$  and  $M = (\{m\}_{K_1}, \{(K_1, K_2)\}_K)$ , and assume an empty initial knowledge. If  $K$  is weaker than  $K_1$ , we have that  $p_{dec}(\{m\}_{K_1}) \leq p_{dec}(\{(K_1, K_2)\}_K)$  and  $pGuess_M(\{K_1\}) = pGuess_M(\{K\}) = p_{dec}(\{(K_1, K_2)\}_K)$ . Therefore, given  $\hat{p} = p_{dec}(\{(K_1, K_2)\}_K)$ , we have  $pP_M = (m_{\cdot\hat{p}}, (K_{1\cdot\hat{p}}, K_{2\cdot\hat{p}})_{\cdot\hat{p}})_{\cdot 1}$ . On the other hand, from the previous examples and from the condition  $p_{dec}(\{m\}_{K_1}) \leq p_{dec}(\{(K_1, K_2)\}_K)$ , we obtain the probabilistic pattern  $pP_N = (m_{\cdot\hat{p}}, (K_{1\cdot\hat{p}}, K_{2\cdot\hat{p}})_{\cdot\hat{p}})_{\cdot 1}$  and, since  $pMax_M = pMax_N = \hat{p}$ , we also obtain  $M \approx N$ . In conclusion, we observe that ciphering the first block  $m$  of  $N$  with both keys  $K_1$  and  $K_2$  is not meaningful, since  $N$  is probabilistically equivalent to an expression where this information is ciphered with one of those keys only. Indeed, an adversary can gain information about  $m$  by cryptanalyzing the second block  $\{(K_1, K_2)\}_K$ .

## 3 Indistinguishability

The notion of equivalence presented above is not adequate to state the indistinguishability among cryptographic expressions. On the one hand, it is not realistic to require that the same ciphered blocks have to be decrypted exactly with the same probability. On the other hand, it is not worth considering those blocks that can be decrypted with negligible probability, since essentially they are *almost* equivalent to a black box.

In order to relax the notion of equivalence for cryptographic expressions, we introduce a relation, called  $\varepsilon$ -*probabilistic similarity* ( $\approx_\varepsilon$ ), which (i) approximates the equivalence by introducing a tolerance to small differences of the probabilistic parameters that are associated with the probabilistic patterns, and (ii) allows for equating the black box and those ciphertexts that can be decrypted with a negligible probability.

Given an initial knowledge  $G$ , we say that  $M$  and  $N$  are  $\varepsilon$ -probabilistically similar ( $M \approx_\varepsilon^G N$ ) if  $pMax_M^G$  and  $pMax_N^G$  are *almost the same* up to the tolerance  $\varepsilon$  and if  $M$  and  $N$  are  $\varepsilon$ -compatible according to the notion of compatibility  $\sim_\varepsilon$  specified below.

**Definition 2.**  $M \approx_\varepsilon^G N \Leftrightarrow pP_M^G \sim_\varepsilon pP_N^G \wedge |pMax_M^G - pMax_N^G| \leq \varepsilon$ .

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

$P_{\cdot p} \sim_\varepsilon Q_{\cdot p'}$	<i>if</i> $p, p' \leq \varepsilon$	$P_{\cdot p}, Q_{\cdot p'} \in \mathbf{pPat}$
$K_{\cdot p} \sim_\varepsilon K_{\cdot p'}$	<i>if</i> $ p - p'  \leq \varepsilon$	$K \in \mathbf{Keys}$
$m_{\cdot p} \sim_\varepsilon m_{\cdot p'}$	<i>if</i> $ p - p'  \leq \varepsilon$	$m \in \mathbf{String}$
$(P_{\cdot p_1}, Q_{\cdot p_2})_{\cdot p_3} \sim_\varepsilon (P'_{\cdot p'_1}, Q'_{\cdot p'_2})_{\cdot p'_3}$	<i>if</i> $ p_3 - p'_3  \leq \varepsilon$	$\wedge$ $P_{\cdot p_1} \sim_\varepsilon P'_{\cdot p'_1} \wedge Q_{\cdot p_2} \sim_\varepsilon Q'_{\cdot p'_2}$ $P_{\cdot p_1}, Q_{\cdot p_2}, P'_{\cdot p'_1}, Q'_{\cdot p'_2} \in \mathbf{pPat}$

Note that two different pieces of information are indistinguishable if they are associated with probabilistic parameters that are smaller than the given tolerance  $\varepsilon$ , i.e. in practice both of them are considered to be a black box.

For instance, according to such a notion of probabilistic similarity, the expressions  $M = \{m\}_K$  and  $N = \{n\}_{K'}$  are indistinguishable if – given  $G = \emptyset$ ,  $pP_M = m_{\cdot p_1}$ ,  $pP_N = n_{\cdot p_2}$ , and a fixed threshold  $\varepsilon$  – the probabilities  $p_1$  and  $p_2$  are equal or smaller than  $\varepsilon$ . Also the expressions  $M = \{m\}_K$  and  $N = \{m\}_{K'}$  are indistinguishable if  $p_1 = p_{dec}(\{m\}_K)$  and  $p_2 = p_{dec}(\{m\}_{K'})$  are similar (even if not exactly the same). In practice, if  $|p_1 - p_2| \leq \varepsilon$ , then  $M \approx_\varepsilon^G N$ .

**Proposition 1.** *Given  $M, N \in \mathbf{Exp}$  it holds that:*

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

*Proof.* See [21].

**Proposition 2.** *Given  $M, N \in \mathbf{Exp}$  it holds that:*

$$M \approx^G N \quad \Leftrightarrow \quad M \approx_0^G N.$$

*Proof.* A trivial consequence of the definition of compatibility relation.

Finally, note that the case  $\varepsilon = 1$  is not considered, since it is trivial to see that in such a case it follows  $\forall M, N \in \mathbf{Exp} \ M \approx_1^G N$ .

## 4 Relating the Probabilistic and the Dolev-Yao Models

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 two expressions are equivalent within a Dolev-Yao model that relies on perfect cryptography if and only if the two expressions are probabilistically similar under the ideal encryption assumption.

## 4.1 Equivalence within Perfect Cryptography

We start by introducing a notion of equivalence for cryptographic expressions that relies on the perfect cryptography assumption. Such a notion is inspired by that defined in [2]. First, we define a variant of the set of expressions, called **Pat**, which does not contain ciphertexts and includes the new symbol  $\otimes$  (representing a ciphertext that cannot be decrypted by the adversary).

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

Intuitively, a *pattern* is an expression that does not contain encrypted terms and that may contain some part that an adversary cannot decrypt. Now, we 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  $T$ .

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

Then, given an initial knowledge  $G$ , we define function  $pat^G(M)$ , which expresses the pattern obtained from an expression  $M$  by exploiting the knowledge  $G$ , as  $pat^G(M) = p(M, initKeys((M, G)))$ . For example, if  $G$  is empty,  $pat^G(\{\{\{K_1\}_{K_2}\}_{K_3}, K_3\}) = (\otimes, K_3)$ .

Finally, given an initial knowledge  $G$ , we say that two expressions are *equivalent* if they yield the same pattern.

**Definition 3.**  $M \cong^G N \Leftrightarrow pat^G(M) = pat^G(N)$ .

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

## 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., [11]).

**Definition 4.** 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 5.** *An encryption scheme is ideal if and only if*

$$\forall \{N\}_K \in \mathbf{Exp}, \forall G \in \mathbf{Exp} : G \not\vdash K, \forall \text{polynomial } q : \exists \eta_0 \in \mathbb{N} \text{ such that} \\ p_{dec}(\{N\}_K, G) \leq \frac{1}{q(\eta)} \quad \forall \eta > \eta_0.$$

As a consequence, if the assumption of ideal encryption holds, from the definition above we also have that  $\forall \mathcal{A}$  and  $\forall G \in \mathbf{Exp} : G \not\vdash K$ :

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

By following an approach also used in [22], we show that a result holding in the perfect cryptography scenario also holds in our probabilistic model (and vice versa) if the ideal encryption assumption is taken.

**Theorem 1.** *Given  $M, N \in \mathbf{Exp}$ , if the assumption of ideal encryption holds for a polynomial  $q$  and a natural  $\eta_0$ , then, for each  $\eta > \eta_0$ :*

$$M \cong^G N \quad \Leftrightarrow \quad M \approx_\varepsilon^G N \quad \forall \varepsilon \in ]\frac{1}{q(\eta)}, 1 - \frac{1}{q(\eta)}[.$$

**Proof.**  $\Rightarrow$ ) *A proof of the statement is in [21].*

$\Leftarrow$ ) *The statement derives by structural induction on the expression  $M$  and by observing that, by hypothesis,  $M \approx_\varepsilon^G N \Rightarrow pP_M^G \sim_\varepsilon pP_N^G$ . In the following, we denote by  $T_M$  the set  $initKeys((M, G))$  and by  $T_N$  the set  $initKeys((N, G))$ .*

1.  $pP_M^G \sim_\varepsilon pP_N^G = K.1 \quad K \in \mathbf{Keys}$   
 $\Rightarrow$   
 $p(M, T_M) = p(N, T_N) = K \Rightarrow pat^G(M) = pat^G(N) \Rightarrow M \cong^G N$
2.  $pP_M^G \sim_\varepsilon pP_N^G = m.1 \quad m \in \mathbf{String}$   
 $\Rightarrow$   
 $p(M, T_M) = p(N, T_N) = m \Rightarrow pat^G(M) = pat^G(N) \Rightarrow M \cong^G N$
3.  $pP_M^G = P.p \sim_\varepsilon Q.p' = pP_N^G \quad p, p' \leq \varepsilon \quad P.p, Q.p' \in \mathbf{pPat}$   
 $\Rightarrow$   
 $p(M, T_M) = p(N, T_N) = \otimes \Rightarrow pat^G(M) = pat^G(N) \Rightarrow M \cong^G N$
4.  $pP_M^G = (pP_M^G(L_1, \emptyset), pP_M^G(L_2, \emptyset)).1 \sim_\varepsilon (pP_N^G(L'_1, \emptyset), pP_N^G(L'_2, \emptyset)).1 = pP_N^G \Rightarrow$   
 $pP_M^G(L_1, \emptyset) \sim_\varepsilon pP_N^G(L'_1, \emptyset) \wedge pP_M^G(L_2, \emptyset) \sim_\varepsilon pP_N^G(L'_2, \emptyset)$   
 $\Rightarrow$  *by induction hypothesis*  
 $p(L_1, T_M) = p(L'_1, T_N) \wedge p(L_2, T_M) = p(L'_2, T_N) \Rightarrow$   
 $p(M, T_M) = (p(L_1, T_M), p(L_2, T_M)) = (p(L'_1, T_N), p(L'_2, T_N)) = p(N, T_N) \Rightarrow$   
 $pat^G(M) = pat^G(N) \Rightarrow M \cong^G N$

*Under the assumption of ideal encryption, the four cases above include all the interesting situations in which two probabilistic patterns are compatible according to  $\sim_\varepsilon$ . In particular, the condition  $|p - p'| \leq \varepsilon$  is always true if  $p, p' \neq 1$ . Indeed, thanks to the ideal encryption assumption stating that  $p, p' \leq \frac{1}{q(\eta)}$ , if  $p, p' \neq 1$ ,*

we have that  $p, p' < \varepsilon$ . Therefore, the three cases  $K_{\cdot p} \sim_\varepsilon K_{\cdot p'}$ ,  $m_{\cdot p} \sim_\varepsilon m_{\cdot p'}$  and  $(P_{\cdot p_1}, Q_{\cdot p_2})_{\cdot p} \sim_\varepsilon (P'_{\cdot p'_1}, Q'_{\cdot p'_2})_{\cdot p'}$  collapse into the case  $P_{\cdot p} \sim_\varepsilon Q_{\cdot p'}$ ,  $p, p' \leq \varepsilon$  (case 3 of the proof of  $\Leftarrow$ )), when  $p, p' \neq 1$ .

Finally, note that we did not consider the cases in which  $P_{\cdot p} \sim_\varepsilon Q_{\cdot p'}$  for some  $P, Q \in \mathbf{pPat}$  with  $p = 1$  ( $p \neq 1$ ) and  $p' \neq 1$  ( $p' = 1$ ). In such cases, the condition  $|1 - p'| \leq \varepsilon$  ( $|1 - p| \leq \varepsilon$ ) does not hold, since, by the ideal encryption assumption,  $p' \leq \frac{1}{q(\eta)}$  ( $p \leq \frac{1}{q(\eta)}$ ) and, by the premises of the theorem,  $\varepsilon < 1 - \frac{1}{q(\eta)}$ . As a consequence, it is impossible to find a case in which  $P_{\cdot p} \sim_\varepsilon Q_{\cdot p'}$  for some  $P, Q \in \mathbf{pPat}$  with  $p = 1$  ( $p \neq 1$ ) and  $p' \neq 1$  ( $p' = 1$ ). ■

## 5 Secrecy against the Computational Adversary

In this section we introduce a notion of secrecy of some information within a given expression. Consider for example the expression  $M = (\{m\}_K, \{n\}'_K)$ . We are interested in evaluating whether the expression  $M$  *maintains* a given secret  $m$  assuming that the expression  $G$  models the actual knowledge of an adversary<sup>6</sup>. In particular, we are also interested in evaluating the degree of secrecy of  $m$  within  $M$ . Intuitively, we observe that a certain secret  $m$  is private in  $M$  if the expression  $N$ , obtained by substituting every occurrence of  $m$  in  $M$  with  $m' \neq m$ , is similar to  $M$ . More formally, we provide the following definition.

**Definition 6.** *Given a knowledge modeled by the expression  $G$ , a certain secret  $\alpha$  such that  $\alpha \in \mathbf{Keys}$  or  $\alpha \in \mathbf{String}$ , a parameter  $\varepsilon \in [0, 1[$  and an expression  $M \in \mathbf{Exp}$  such that  $\alpha$  occurs in  $M$ , we say that  $\alpha$  is  $\varepsilon_G$ -secret in  $M$  iff the expression  $N$  obtained by substituting every occurrence of  $\alpha$  in  $M$  with the key  $K \neq \alpha$  (if  $\alpha \in \mathbf{Keys}$ ) or with the string  $m \neq \alpha$  (if  $\alpha \in \mathbf{String}$ ) is such that  $M \approx_\varepsilon^G N$ .*

Intuitively, Definition 6, inspired by [1], states that a certain secret  $\alpha$  is private within an expression  $M$  if an adversary is not able to distinguish  $M$  from the expression  $N$  obtained by substituting in  $M$  every occurrence of  $\alpha$  with  $\alpha' \neq \alpha$ . In a sense, that means the adversary is not able to extract  $\alpha$  from  $M$  with a probability greater than  $\varepsilon$ . Therefore, if  $\alpha$  is  $\varepsilon_G$ -secret in  $M$ , we can deduce that the adversary with knowledge  $G$  can extract  $\alpha$  from  $M$  with a success probability equal or smaller than  $\varepsilon$ .

*Example 2.* Consider the expression  $M = (\{m\}_{K_1}, \{K\}_{K_2})$  and a knowledge  $G = K_1$ .

On the one hand, we obviously have that  $m$  is not  $\varepsilon_G$ -secret in  $M$  for any  $\varepsilon \in [0, 1[$ . Given  $N = (\{m'\}_{K_1}, \{K\}_{K_2})$  we have that  $M \not\approx_\varepsilon^G N$ , in fact  $pP_M^G = (m_{\cdot 1}, K_{\cdot p})$  where  $p = p_{dec}(\{K\}_{K_2}, (M, G)) = p_{dec}(\{K\}_{K_2}, (N, G))$  and  $pP_N^G = (m'_{\cdot 1}, K_{\cdot p})$ . Since  $m \neq m'$  we obviously have that  $pP_M^G \not\approx_\varepsilon pP_N^G$  so that, in practice,  $M \not\approx_\varepsilon^G N$ .

<sup>6</sup> The expression  $G$  models, for example, the sequence of messages exchanged within the network until a certain moment, and the set of keys known by the adversary.

On the other hand, we have that  $K$  is  $\varepsilon_G$ -secret in  $M$  for any  $\varepsilon \in [p, 1[$ . Given  $N = (\{m\}_{K_1}, \{K'\}_{K_2})$  we have that  $M \approx_\varepsilon^G N$  for any  $\varepsilon \in [p, 1[$ , in fact  $pP_M^G = (m_{.1}, K_{.p})$  and  $pP_N^G = (m_{.1}, K'_{.p})$ . Since  $\varepsilon \geq p$  we have that the adversary is not able to distinguish  $K$  from  $K'$  ( $pP_M^G \sim_\varepsilon pP_N^G$ ), so that in practice we have that  $M \approx_\varepsilon^G N$ .

## 5.1 An Application

We apply the notion of secrecy within an expression in a very simple real case. Consider a protocol where a server  $S$  could be asked to generate a secret key and then send it back to the entity  $A$  that applied the request. The server also monitors and keeps track of all the messages exchanged in the network.

Assuming that an authentication phase precedes the protocol, we denote with  $K_{SA}$  a key shared between the server  $S$  and the entity  $A$ . Finally, we use  $t$  to denote a time stamp. The protocol can be described as follows (with the standard notation  $A \rightarrow B : Msg$  we denote a message  $Msg$  sent by  $A$  and received by  $B$ ):

1.  $A \rightarrow S : \{request, A, S, t\}_{K_{SA}}$
2.  $S \rightarrow A : \{K, S, A, t\}_{K_{SA}}$

where  $K$  is the secret key generated by the server.

We now translate the messages exchanged by the protocol into cryptographic expressions, by assuming that  $K_{SA}, K \in \mathbf{Keys}$  and that  $request, A, S, t$  correspond to  $r, a, s, t \in \mathbf{String}$ , respectively. Hence, we have that the protocol exchanges the following expressions:

1.  $\{(r, ((a, s), t))\}_{K_{SA}}$
2.  $\{(K, ((s, a), t))\}_{K_{SA}}$

Now, assume that all the messages exchanged in the network are modeled by the formal expression  $G$ . Then, we apply our notion of secrecy within expressions in order to check whether the expression  $\{(K, ((s, a), t))\}_{K_{SA}}$  ensures a given degree  $\varepsilon$  of secrecy for  $K$ . To this end, what the server needs to do is to check whether  $K$  is  $\varepsilon_G$ -secret in  $\{(K, ((s, a), t))\}_{K_{SA}}$ . The parameter  $\varepsilon$  is fixed by the server and represents the security threshold needed to guarantee the secure execution of the protocol. Note that, as the traffic of information within the network increases and the amount of messages ciphered with the shared key  $K_{SA}$  gets larger, the server may not guarantee the  $\varepsilon_G$ -secrecy anymore. Our notion of secrecy within expressions is able to capture situations of this kind. Therefore, if at a certain instant of time  $K$  is not  $\varepsilon_G$ -secret in  $\{(K, ((s, e), t))\}_{K_{SA}}$  anymore, the server may, for example, activate a procedure generating a new shared key with the entity  $A$  and then send the secret to  $A$  encrypted with the fresh key.

## 6 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 [14] where, in the setting of a subset of asynchronous  $\pi$ -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 [5] 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 [13] on the basis of a simple model where public outputs are observed in order to infer the content of secret inputs. Finally, [22] 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. Moreover, in [22] the authors define a function similar to our  $p_{dec}$  in order to model the probability for a computational adversary of guessing a key. However they abstract away from the particular ciphertext in which the key to be guessed is used as the ciphering key, and from the knowledge the adversary gets. As we do, they also abstract away from how the probability  $p_{dec}(\{m\}_K)$  could be computed.

We finally point out that probabilistic notions of security as well as approximate security properties can be found in the literature (see, e.g., [10, 7, 4, 3]), but they do not relate probability and cryptographic primitives.

## 7 Conclusion

In this paper we proved that a standard notion of Dolev-Yao adversary equates the expressive power of a computational adversary in the case ideal encryption is assumed. This is done in a formal framework where indistinguishability among cryptographic expressions is defined by means of a notion of probabilistic similarity taking into account computational poly-time adversaries.

The formal comparison among cryptographic expressions and its application to the verification of security properties represents an important step towards the definition of a formal framework for modelling cryptographic protocols and analyzing their robustness against malicious parties. In particular, the robustness of a system can be evaluated both in terms of absence of (probabilistic) covert channels and in terms of effectiveness of cryptanalysis attacks. While the first

kind of attack has been deeply analyzed in the literature (see, e.g., [7, 4] and the references therein), in this paper we concentrated on the second type of security problem and we proposed an approach, which, as a further line of investigation, aims at putting the basis for a formal framework where both families of security flaws can be attacked in an integrated way.

## References

1. M. Abadi, A. D. Gordon, “*A Calculus for Cryptographic Protocols: The Spi Calculus*”, Information and Computation, 148(1):1–70, 1999.
2. 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.
3. 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.
4. A. Aldini, M. Bravetti, R. Gorrieri, “*A Process-algebraic Approach for the Analysis of Probabilistic Non-interference*”, Journal of Computer Security, vol. 12(2):191-245, IOS Press, 2004.
5. M. Backes, B. Pfizmann, “*Computational Probabilistic Non-interference*”, in Proc. of 7th European Symposium on Research in Computer Security, Springer LNCS 2502:1-23, 2002.
6. A. Datta, R. Kusters, J. C. Mitchell, A. Ramanathan, V. Shmatikov, “*Unifying Equivalence-Based Definitions of Protocol Security*”, in Proc. of Workshop on Issues in the Theory of Security, WITS’04, 2004.
7. 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.
8. D. Dolev, A. Yao, “*On the Security of Public-key Protocols*”, IEEE Transactions on Information Theory, 29:198-208, 1983.
9. 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.
10. J. W. Gray III, “*Toward a Mathematical Foundation for Information Flow Security*”, Journal of Computer Security, 1:255-294, 1992.
11. 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.
12. R. A. Kemmerer, “*Analyzing Encryption Protocols using Formal Verification Techniques*”, IEEE Journal on Selected Areas in Communications, 7(4):448-457, 1989.
13. 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.
14. 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.
15. D. Micciancio, B. Warinschi, “*Completeness Theorems for the Abadi-Rogaway Language of Encrypted Expressions*”, in 2nd ACM SIGPLAN and IFIP WG 1.7 Workshop on Issues in the Theory of Security (WITS’02), Portland (OR), 2002.

16. 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.
17. L. C. Paulson, “*The Inductive Approach to Verifying Cryptographic Protocols*”, Journal of Computer Security, 6(1-2):85-128, 1998.
18. A. Ramanathan, J. Mitchell, A. Scedrov, V. Teague, “*Probabilistic Bisimulation and Equivalence for Security Analysis of Network Protocols*”, in Proc. of 7th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS’04), Springer LNCS 2987:468-483, 2004.
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.
21. A. Troina, A. Aldini, R. Gorrieri, “*Approximating Imperfect Cryptography in a Formal Model*”, in Proc. of Mefisto Project Final Workshop, Elsevier ENTCS, to appear, available at <http://mefisto.web.cs.unibo.it/pubbl.html>.
22. R. Zunino, P. Degano, “*A Note on the Perfect Encryption Assumption in a Process Calculus*”, in Proc. of 7th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS’04), Springer LNCS 2987:514-528, 2004.