A Probabilistic Formulation of Imperfect Cryptography

Angelo Troina[1], Alessandro Aldini[2], and Roberto Gorrieri[3]

[1] Dipartimento di Informatica, University of Pisa,
troina@di.unipi.it
[2] Istituto STI, University of Urbino,
aldini@sti.uniurb.it
[3] Dipartimento di Scienze dell'Informazione, University of Bologna,
gorrieri@cs.unibo.it

Abstract. We present a novel equivalence for cryptographic expressions that overcomes two limitations of classical security models: perfect cryptography and nondeterministic adversary. The uncertainty concerning the robustness of cryptographic primitives against breaking attacks is estimated through probabilistic information. We also define an approximated relation that allows cryptographic expressions (leading the same information) that can be broken with similar probabilities to be indistinguishable from the adversary viewpoint. Finally, we show that the equivalence is preserved when passing to the usual Dolev-Yao model.

1 Introduction

The use of formal methods for modeling and analyzing cryptographic operations is well-established. Since the seminal paper by Dolev and Yao [6] 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., [4, 10, 8, 7, 12, 11]). Key to success of such a theory was the very simple idea behind the definition of a ciphertext. In practice, a message encrypted with a given key K can be decrypted and read only by someone knowing the key K, while for each other user such a message is a black box lacking in any sense. For instance, a recent formal view of cryptography introduced by Abadi and Rogaway [1] defines formal algebraic cryptographic expressions and a related notion of equivalence. The novelty is that such an approach relates the formal view and the classical computational model of cryptography, by proving as the main result the soundness of the formal world with respect to the computational world. The reason for relating these two views is that the simplicity of the formal model, which deals with formal expressions of an abstract algebra, is opposed to the concreteness of the computational model, which deals with probability and computational complexity of algorithms.

Two assumptions that make the formal view quite disconnected from the real setting of cryptographic algorithms are the perfect cryptography hypothesis and the nondeterministic adversary model. The former assumption says that

$\{M\}_K$ (representing the encryption of M with the key K) and $\otimes$ (representing an undecryptable ciphertext) are always equivalent if the key K is not known. The latter constraint says that an adversary is not allowed to learn secrets by guessing them. However, the robustness of a ciphertext may be jeopardized by the several environment variables, such as clever attackers that may succeed in retrieving (some bits of) information (e.g., by randomly guessing data, or by analyzing a large amount of ciphertext, or by employing a partial knowledge of the plaintext, or by breaking weak keys and too simple, foreseeable cryptographic algorithms). In practice, it is not possible to obtain the guarantee of absolute secrecy, which instead is assumed by the usual formal models of cryptographic operations.

In this paper, we overcome the two limitations mentioned above in the following way. As far as the adversary model is concerned, we interpret intruders as probabilistic processes that may randomly guess data, perform statistical analysis of exchanged information, and use partial information to reduce the range of exhaustive searches. On the basis of such an adversary and of its knowledge, we assume that a bound on the ability of obtaining information from a particular ciphertext is estimated and passed as a probabilistic parameter to the algorithm that decides the equivalence between cryptographic expressions. More formally, we employ a function parameterized by the initial knowledge of the adversary, whose outcome is strictly related to the considerations surveyed above and to other aspects, such as the expected robustness of the involved keys and of the particular ciphering algorithm, which can be subject to cryptanalysis attempts. Such an outcome represents an estimation of the probability of obtaining useful information from a given ciphertext, without knowing the related key. For instance, expression $(\{M\}_K, N)$, which expresses a pair given by a ciphertext and a plaintext not including information about the key K, may easily reveal information about M in case K is a very short key, the used algorithm is a stream cipher, and the initial knowledge of the adversary includes 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 is close to 1. On the other hand, if K is an adequately long key and the ciphering algorithm is considered to be robust against any known attack, then the probability of retrieving M is at most equal to the probability of randomly guessing K.

When computing the probability of retrieving data, the knowledge of the adversary increases as he succeeds in obtaining new information. Therefore, the estimation of the adversary ability of capturing confidential data always depends on the knowledge he may collect step by step. On the other hand, one may claim that the estimation of the adversary ability of breaking secrets cannot be accurate because, e.g., the effectiveness of unknown attacks cannot be evaluated, or the information available about the robustness of each cryptographic ingredient is not enough to quantify the success probability of any attack. As a consequence, the distinction between sets of cryptographic expressions in practice may be too strong. To this aim, the closure among cryptographic expressions can be approximated by introducing a ε-tolerance, which permits those expressions that require almost the same effort to reveal the same information to be not distinguished by the adversary.

The model we employ is inspired by the formal encryption framework defined by Abadi and Rogaway. However, in our setting, two cryptographic expressions turn out to be probabilistically equivalent if they yield the same information and this information can be obtained with the same probability. Therefore, we abandon the usual Dolev-Yao abstraction and we take into consideration cryptanalysis attacks. Anyway, we point out that if we abstract away from probabilities we obtain again a model that is compliant with the Dolev-Yao based formal encryption framework.

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, as shown, e.g., in [3] in the context of a model of asynchronous probabilistic reactive systems, and in [9], which employs an asymptotic notion of probabilistic equivalence that takes into consideration polynomial time attacks. Moreover, probabilistic notions of security as well as approximated security properties can be found in the recent literature (see, e.g., [7, 5, 2]), but they do not relate probability and cryptographic primitives.

2 Background

In this section, we recall the formal view of cryptography presented in [1]. In particular, expressions are built up from bits and keys by pairing and encryption. The equivalence relation captures when two expressions *look the same* from the viewpoint of an adversary that has no prior knowledge of the keys used within the expressions.

2.1 Expressions

We use **Bool** for the set of bits {0,1}. Bits are used to express numbers and principal names. We use **Keys** to denote a fixed, nonempty set of symbols disjoint from **Bool**. In the following, the symbols $K, K', K'', \ldots$ and $K_1, K_2, K_3, \ldots$ range over the set **Keys**. We use **Exp** to denote the set of *expressions* defined by the grammar:

$M, N ::=$	expressions
K	key (for $K \in$ **Keys**)
i	bit (for $i \in$ **Bool**)
(M, N)	pair
$\{M\}_K$	encryption (for $K \in$ **Keys**)

Informally, (M, N) represents the pairing of M and N, while $\{M\}_K$ represents the encryption of M under K via a symmetric encryption algorithm (like, e.g., 3DES). Pairing and encryption can be nested, like, e.g., in the expression $(\{\{(0, K)\}_{K_1}\}_{K_2}, K_1)$.

2.2 Equivalence

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

$$\boxed{\begin{array}{ll}
M \mapsto 0 \ \wedge \ M \mapsto 1 & \\
M \mapsto M & \\
M \mapsto N_1 \ \wedge \ M \mapsto N_2 & \Rightarrow M \mapsto (N_1, N_2) \\
M \mapsto (N_1, N_2) & \Rightarrow M \mapsto N_1 \ \wedge \ M \mapsto N_2 \\
M \mapsto N \ \wedge \ M \mapsto K & \Rightarrow M \mapsto \{N\}_K \\
M \mapsto \{N\}_K \ \wedge \ M \mapsto K & \Rightarrow M \mapsto N
\end{array}}$$

$M \mapsto N$ expresses what an attacker can obtain from M without any prior knowledge of the keys used in M. For example, we have $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \mapsto K_3$, and $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \mapsto \{K_1\}_{K_2}$, but $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \not\mapsto K_1$.

Then, we define the set **Pat** of *patterns* as an extension of the set of expressions that employs the new symbol $\otimes$ representing a ciphertext that an attacker cannot decrypt.

$$\boxed{\begin{array}{lll}
P, Q ::= & & \text{patterns} \\
& K & \quad \text{key (for } K \in \mathbf{Keys}\text{)} \\
& i & \quad \text{bit (for } i \in \mathbf{Bool}\text{)} \\
& (P, Q) & \quad \text{pair} \\
& \{P\}_K & \quad \text{encryption (for } K \in \mathbf{Keys}\text{)} \\
& \otimes & \quad \text{undecryptable text}
\end{array}}$$

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

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

Moreover, we define $pattern(M)$, which expresses the pattern obtained from an expression M without knowing a priori any auxiliary set T of keys. Formally, $pattern(M) = p(M, \{K \in \mathbf{Keys} \mid M \mapsto K\})$. For example, we have: $pattern((\{\{K_1\}_{K_2}\}_{K_3}, K_3)) = (\{\otimes\}_{K_3}, K_3)$.

Finally, we say that two expressions are *equivalent* if they yield the same pattern:

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

For example, we have $(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \cong (\{\{0\}_{K_1}\}_{K_3}, K_3)$ since both expressions yield the pattern $(\{\otimes\}_{K_3}, K_3)$.

3 Probabilistic equivalence

We now introduce an equivalence relation that takes into account the possibility for an attacker of obtaining plaintext from an expression $\{M\}_K$ without knowing the key K. We abandon so the assumption of perfect cryptography, commonly used in the body of literature that treats cryptographic operations in purely formal models. For our purpose, we give a new definition for patterns, which were used to denote the information (associated to a ciphertext) employed to decide the equivalence between expressions. Here, we define the set **pPat** of *probabilistic patterns* as an extension of the set of expressions, with the grammar:

$P, Q ::=$	probabilistic patterns
K	key (for $K \in \mathbf{Keys}$)
i	bit (for $i \in \mathbf{Bool}$)
(P, Q)	pair
$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
$P._p$	probabilistic expression (for $P._p \in \mathbf{pExp}$)

Intuitively, a probabilistic pattern is an expression that may contain some parts that an attacker can decrypt with a certain probability. A *probabilistic expression* $P._p$ represents an expression that does not contain undecryptable blocks and is associated with a probabilistic parameter $p \in]0, 1]$, which represents the probability of getting in clear plaintext contained in P. Formally, we define the set **pExp** of *probabilistic expressions* with the grammar:

$P._p, Q._p ::=$	probabilistic expressions
$K._p$	key (for $K \in \mathbf{Keys}$)
$i._p$	bit (for $i \in \mathbf{Bool}$)
$(P._p, Q._p)._p$	pair
$p \in]0, 1]$	

A probabilistic expression associated to a ciphertext is obtained by substituting every ciphered block with the corresponding expression in clear associated with the probability of obtaining it. Given an adversary described by an algorithm $\mathcal{A}$ with initial knowledge modeled by expression G, the probabilistic pattern associated with the expression $\{0\}_K$ (such that $G \not\mapsto K$) is defined by $0.p_{dec}(\{0\}_K, G, \mathcal{A})$. The function $p_{dec}(\{N\}_K, G, \mathcal{A})$ calculates the probability for $\mathcal{A}$ of obtaining useful information from the ciphertext $\{N\}_K$ by exploiting the knowledge G. The outcome of the function p_{dec} depends on many factors, such as the computational power of (and the information collected by) $\mathcal{A}$, the expected robustness of the key K against guesses or attacks, and the particular ciphering algorithm. In the rest of the paper, we abstract away from the computational power of the adversary and we omit parameter $\mathcal{A}$. Function p_{dec}, whose outcome denotes the robustness of a ciphertext against cryptanalysis attempts, is not sufficient to define the probability of decrypting a ciphered block. Consider, for example, the expression $(\{\{0\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K)$. What is the probability of getting the bit 0 in clear? A simple and immediate answer could be $p_{dec}(\{\{0\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{0\}_{K_1})$ [4], which is the probability of guessing the two keys K_2 and K_1 (i.e., by breaking the block $\{\{0\}_{K_1}\}_{K_2}$ and then by breaking the remaining block $\{0\}_{K_1}$). However, by analyzing the expression, we may observe that an attacker can obtain the two keys K_1 and K_2, needed to get the bit 0 in clear, by breaking the second block with a probability equal to $p_{dec}(\{(K_1, K_2)\}_K)$. Therefore, the probability of breaking a block may vary according to the strategy an attacker uses when he tries to cryptanalyze an expression. Obviously, we have to assume that an attacker follows the optimal approach, and so we always associate to a ciphered block the maximum probability of getting it in clear. For this reason, we have to analyze all possible cryptanalysis paths that an attacker can follow. This kind of analysis is done by employing several auxiliary structures and functions, which we informally introduce as follows.

Given an expression M and an adversary with 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 guessing the keys contained in T by following a particular strategy. By employing the probabilistic values contained in the set $\mathbf{pKeys}_M^G$, we get two kinds of information. On the one hand, for each set T of keys (which can be derived from M) we compute $pGuess_M^G(T)$, which is the maximum probability of guessing the 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 plaintext contained in the expression M. Then, we use a function $pExp_M^G$, which employs the results obtained by applying the function $pGuess_M^G(T)$ in order to turn pieces of the expression M into probabilistic expressions. Finally, we employ a function pP_M^G which turns the expression M into a probabilistic pattern by converting each undecryptable ciphertext into a probabilistic expression through the function $pExp_M^G$. We formally detail all these structures in the next subsections.

The novel equivalence relation we model captures when two messages contain the same information and this information is obtained with the same probability in case an attacker tries to cryptanalyze ciphered pieces of data. Formally, we verify that two expressions M and N are equivalent if they yield the same probabilistic pattern (obtained through the functions pP_M^G and pP_N^G) and if the probabilities of getting in clear the expression (expressed by the parameters $pMax_M^G$ and $pMax_N^G$) are equal. In the rest of the paper, the initial knowledge G of the adversary, which is a parameter needed by function p_{dec}, is assumed to be the expression $(0, 1)$ and we omit it when it is clear from the context. Obviously, it is worth noting that as the attacker acquires additional information, the enriched knowledge of the adversary may be responsible for increasing the estimation given by the function p_{dec}.

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

3.1 pKeys

The first step of our procedure, which allows us to turn an expression M into a probabilistic pattern, consists of generating the set $\mathbf{pKeys}_M^G$ containing the elements (T, p)[5] such that $T \subseteq \mathbf{Keys}$ is a set of keys recoverable from the expression (M, G) through any cryptanalysis strategy, and $p \in]0, 1]$ is the probability of guessing all the keys in T. The set $\mathbf{pKeys}_M^G$ is generated by the following two-step algorithm:

```
$\mathbf{pKeys}_M^G = \{(initKeys((M, G)), 1)\}$;
$addKeys((M, G), 1)$;
```

where $initKeys : \mathbf{Exp} \rightarrow \mathcal{P}(\mathbf{Keys})$ takes in 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:

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

Initially, $\mathbf{pKeys}_M^G$ is the set $(initKeys((M, G)), 1)$, where, as we have seen, $initKeys((M, G)) \subseteq \mathbf{Keys}$ is the set of keys that can be derived from the expressions M and G with probability 1 (i.e., the keys an attacker infers from the expression M and from his initial knowledge G without cryptanalysis attempts). In particular, $initKeys((M, G))$ contains all keys K such that $(M, G) \mapsto K$.

At each step, we add to $\mathbf{pKeys}_M^G$ sets of keys recoverable from the expression M by breaking the ciphered blocks. In particular, for each cryptanalysis strategy that an attacker may follow, $\mathbf{pKeys}_M^G$ contains the set of keys guessed by following that strategy and the probability of guessing such keys. The procedure $addKeys$ recursively adds to the set $\mathbf{pKeys}_M^G$ the results of each possible cryptanalysis strategy.

Note that the function $p_{dec}(\{N\}_K, H)$ (with $N \in \mathbf{Exp}, K \in \mathbf{Keys}$) calculates the probability, for an adversary with knowledge H, of getting the expression N in clear without knowing the key K. In particular, G represents the initial knowledge of the adversary, and H represents the knowledge obtained by intercepting and analyzing the expression M.

Example 1. Given $M = (((\{\{(0, K)\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K), K')$, $\mathbf{pKeys}_M$ is initialized with the keys inferred from (M, G) with probability 1 through the entailment relation. Since we assumed that the adversary has no initial knowledge, we just have $M \mapsto K'$, then we start with $\mathbf{pKeys}_M = \{\{K'\}._1\}$. Then, the $addKeys$

[5] In the following we use the abbreviation $T._p$ to stand for (T, p).

procedure evaluates all possible cryptanalysis sequences an attacker may follow, by adding at each step new elements to the set $\mathbf{pKeys}_M$ (see Fig. 1). We observe that the set of keys $\{K', K, K_1, K_2\}$ appears three times in the set $\mathbf{pKeys}_M$ (see Fig. 2) with different probabilities. This is due to the alternative strategies an attacker may follow to decide the order in which he tries to obtain the plaintext.

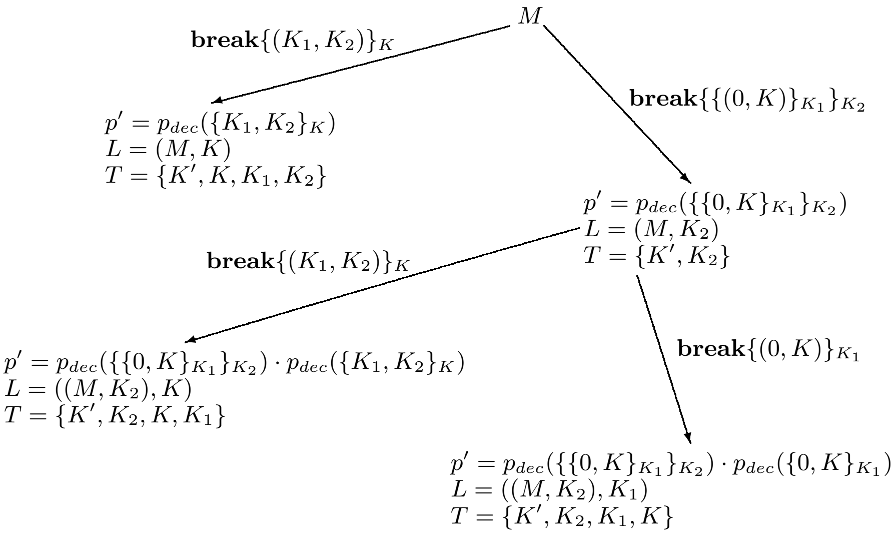
M
$\mathbf{break}\{(K_1, K_2)\}_K$
$\mathbf{break}\{\{(0, K)\}_{K_1}\}_{K_2}$
$p' = p_{dec}(\{K_1, K_2\}_K)$
$L = (M, K)$
$T = \{K', K, K_1, K_2\}$
$p' = p_{dec}(\{\{0, K\}_{K_1}\}_{K_2})$
$L = (M, K_2)$
$T = \{K', K_2\}$
$\mathbf{break}\{(K_1, K_2)\}_K$
$\mathbf{break}\{(0, K)\}_{K_1}$
$p' = p_{dec}(\{\{0, K\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{K_1, K_2\}_K)$
$L = ((M, K_2), K)$
$T = \{K', K_2, K, K_1\}$
$p' = p_{dec}(\{\{0, K\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{0, K\}_{K_1})$
$L = ((M, K_2), K_1)$
$T = \{K', K_2, K_1, K\}$

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

$$\mathbf{pKeys}_M = \left\{ \begin{array}{l} \{K'\}_{\cdot 1}, \\ \{K', K, K_1, K_2\}_{\cdot p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_2\}_{\cdot p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2})}, \\ \{K', K_2, K, K_1\}_{\cdot p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_2, K_1, K\}_{\cdot p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(0,K)\}_{K_1})} \end{array} \right\}$$

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

3.2 *pGuess*

The function $pGuess^G_M(T)$ computes the maximum probability for the attacker of guessing all keys in the set T according to the best cryptanalysis strategy he may follow to decipher the expression M.

$pGuess^G_M : \mathcal{D}_{pGuess^G_M} \to]0,1]$, with $\mathcal{D}_{pGuess^G_M} = \{T \in \mathcal{P}(\mathbf{Keys}) \mid \exists J.p \in \mathbf{pKeys}^G_M : T \subseteq J\}$, is formally defined as:

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

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

Example 2. Consider again $M = ((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. We have that (see the set $\mathbf{pKeys}_M$ in Fig. 2):

$$\begin{aligned}
pGuess_M(\{K'\}) &= 1 \\
pGuess_M(\{K_2\}) &= \max\left\{\begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(0,K)\}_{K_1}) \end{array}\right\} \\
&= \max\left\{\begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \end{array}\right\} \\
pGuess_M(\{K\}) &= \max\left\{\begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(0,K)\}_{K_1}) \end{array}\right\} \\
pGuess_M(\{K',K,K_1,K_2\}) &= \max\left\{\begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2} \cdot p_{dec}(\{(0,K)\}_{K_1}) \end{array}\right\}
\end{aligned}$$

3.3 *pMax*

Given an expression M, the parameter $pMax^G_M$ expresses the probability of getting in clear all the information contained in M. Therefore, $pMax^G_M$ represents the maximum probability of guessing all keys used in M. Let $allKeys(M)$ be the set of all key symbols that occur in M; $pMax^G_M$ is derived in the following way:

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

If we consider the definitions of $pMax^G_M$ and $pGuess^G_M$, we observe that $pMax^G_M$ can also be seen as:

$$pMax^G_M = pGuess^G_M(allKeys(M)).$$

Example 3. Consider the expression $M = ((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. Since we have $allKeys(M) = \{K',K,K_1,K_2\}$, it follows that (see the values of $pGuess_M$ in Example 2):

$$\begin{aligned}
pMax_M &= pGuess_M(\{K',K,K_1,K_2\}) \\
&= \max\left\{\begin{array}{l} p_{dec}(\{(K_1,K_2)\}_K), \\ p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2}) \cdot p_{dec}(\{(0,K)\}_{K_1}) \end{array}\right\}
\end{aligned}$$

3.4 *pExp*

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

The function $pExp^G_M : \mathbf{Exp} \times \mathcal{D}_{pGuess^G_M} \rightarrow \mathbf{pExp}$ is formally defined as follows:

$$\begin{array}{lll}
pExp^G_M(K,T) & = K._{pGuess^G_M(T)} & (K \in \mathbf{Keys}) \\
pExp^G_M(i,T) & = i._{pGuess^G_M(T)} & (i \in \mathbf{Bool}) \\
pExp^G_M((N_1,N_2),T) & = (pExp^G_M(N_1,T), pExp^G_M(N_2,T))._{pGuess^G_M(T)} & \\
pExp^G_M(\{N\}_K,T) & = pExp^G_M(N,T') & (T' = T \cup \{K\})
\end{array}$$

The probabilistic expression that can be obtained from an expression N through the best cryptanalysis strategy applied to M is $pExp^G_M(N,\emptyset)$.

Example 4. Consider again $M = ((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K')$. We have that:
$pExp_M(M,\emptyset) = pExp_M(((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K'), \emptyset) =$
$(pExp_M((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), \emptyset), K'._1)._1$
where from
$pExp_M((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), \emptyset)$
we get
$pExp_M(\{\{(0,K)\}_{K_1}\}_{K_2}, \emptyset) = (0._p, K._p)._p$, given $p = pGuess_M(\{K_2, K_1\})$
and we get
$pExp_M(\{(K_1,K_2)\}_K, \emptyset) = (K_1._{p'}, K_2._{p'})._{p'}$, given $p' = pGuess_M(\{K\})$.

3.5 *pP*

The family of functions *pP* turns expressions into probabilistic patterns. In particular, pP^G_M takes a pair (N,T) (with $N \in \mathbf{Exp}$ and $T \subseteq \mathbf{Keys}$) and, by employing the function $pExp^G_M$, converts all data contained in N and encrypted with unknown (not in T) keys into probabilistic expressions, by associating to such data the probability of obtaining them.

The function $pP^G_M : \mathbf{Exp} \times \mathcal{P}(\mathbf{Keys}) \rightarrow \mathbf{pPat}$ is formally defined as follows:

$$\begin{array}{ll}
pP^G_M(K,T) & = K \quad \text{(for } K \in \mathbf{Keys}) \\
pP^G_M(i,T) & = i \quad \text{(for } i \in \mathbf{Bool}) \\
pP^G_M((N_1,N_2),T) & = (pP^G_M(N_1,T), pP^G_M(N_2,T)) \\
pP^G_M(\{N\}_K,T) & = \begin{cases} \{pP^G_M(N,T)\}_K & \text{if } K \in T \\ pExp^G_M(\{N\}_K,\emptyset) & \text{otherwise} \end{cases}
\end{array}$$

Finally, the probabilistic pattern that can be obtained from the expression M (with initial knowledge G) is:

$$pP_M^G(M, initKeys((M, G))).$$

In the following, given an expression M, we use the abbreviation pP_M^G (with no arguments) to stand for $pP_M^G(M, initKeys((M, G)))$.

Example 5. Given the expression $M = ((\{\{(0, K)\}_{K_1}\}_{K_2}, \{(K_1, K_2)\}_K), K')$, we have that:

$$pP_M = ((pExp_M(\{\{(0, K)\}_{K_1}\}_{K_2}, \emptyset), pExp_M(\{(K_1, K_2)\}_K, \emptyset)), K').$$

Example 6. Another interesting example we want to propose is that of two expressions that yield the same probabilistic patterns but have two different $pMax$ values. Consider the following expressions:

$$M = (\{0\}_K, \{1\}_K) \quad N = (\{0\}_K, \{1\}_{K'}), \quad K \neq K'.$$

We have that:

$$pP_M = (0._{\hat{p}}, 1._{\hat{p}}),$$

where $\hat{p} = pGuess_M(\{K\}) = \max\{p_{dec}(\{0\}_K), p_{dec}(\{1\}_K)\}$. The intuition is that an attacker can get information contained in the expression M by guessing the key K used to cipher both blocks. On the other hand, if $pGuess_M(\{K\}) = pGuess_N(\{K\}) = pGuess_N(\{K'\})$ we also have that:

$$pP_N = (0._{\hat{p}}, 1._{\hat{p}}) = pP_M.$$

Hence, M and N have the same probabilistic patterns, even if to get in clear the whole expression N an attacker 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, the value $pMax$ 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.

3.6 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 they contain the same information and this information is obtained with the same probability in case it is ciphered with unknown keys.

Example 7. Consider the expressions:

$$\begin{aligned} M &= ((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K') \\ N &= ((\{(0,K)\}_{K_1}, \{(K_1,K_2)\}_K), K') \ . \end{aligned}$$

On the one hand, we have that:

$$\mathbf{pKeys}_N = \left\{ \begin{array}{l} \{K'\}._1, \\ \{K', K, K_1, K_2\}._{p_{dec}(\{(K_1,K_2)\}_K)}, \\ \{K', K_1, K, K_2\}._{p_{dec}(\{(0,K)\}_{K_1})} \end{array} \right\}$$

so if $p_{dec}(\{(0,K)\}_{K_1}) \leq p_{dec}(\{(K_1,K_2)\}_K)$ 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

$$pP_N = (((0._{\hat{p}}, K._{\hat{p}})._{\hat{p}}, (K_1._{\hat{p}}, K_2._{\hat{p}})._{\hat{p}}), K').$$

On the other hand, from Example 1 and the following ones we also derive: $pP_M = (((0._{\hat{p}}, K._{\hat{p}})._{\hat{p}}, (K_1._{\hat{p}}, K_2._{\hat{p}})._{\hat{p}}), K')$ and, since $pMax_M = pMax_N = \hat{p}$, we also have $M \approx N$. In conclusion, we observe that it is useless in the expression M to cipher the first block $(0, K)$ with both keys K_1 and K_2, 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 the data in clear just by breaking the second block $\{(K_1,K_2)\}_K$.

Example 8. Consider an adversary with knowledge $G = K_1$ and again the expressions:

$$\begin{aligned} M &= ((\{\{(0,K)\}_{K_1}\}_{K_2}, \{(K_1,K_2)\}_K), K') \\ N &= ((\{(0,K)\}_{K_1}, \{(K_1,K_2)\}_K), K') \ . \end{aligned}$$

It is easy to see 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:

$$\begin{aligned} \mathbf{pKeys}^G_N &= \{\{K', K_1, K, K_2\}._1\} \\ \mathbf{pKeys}^G_M &= \left\{ \begin{array}{l} \{K', K_1\}._1, \\ \{K', K_1, K_2, K\}._{p_{dec}(\{\{(0,K)\}_{K_1}\}_{K_2})}, \\ \{K', K_1, K, K_2\}._{p_{dec}(\{(K_1,K_2)\}_K} \end{array} \right\} \end{aligned}$$

4 Conservative Extension

In this section, we show that if two expressions are equivalent in the imperfect cryptography scenario, then they are still equivalent if we assume the usual Dolev-Yao model. In practice, our probabilistic equivalence ($\approx$) is a conservative extension of the Abadi-Rogaway equivalence ($\cong$). Intuitively, in our setting the blocks that the Abadi-Rogaway equivalence considers as undecryptable text are opened via cryptanalysis attacks. If two expressions yield the same probabilistic pattern, then they have the same probabilistic expressions associated with the undecryptable text in the Abadi-Rogaway view, and so they yield the same pattern.

Theorem 1 $M, N \in \mathbf{Exp} \quad M \approx N \Rightarrow M \cong N$
Proof *The statement derives by structural induction on the expression M and by observing that, by hypothesis, $M \approx N \Rightarrow pP_M = pP_N$. In the following, we denote by T_M the set $initKeys(M)$ and by T_N the set $initKeys(N)$.*

1. $pP_M = pP_N = i \qquad i \in \mathbf{Bool}$
   $\Rightarrow$
   $p(M, T_M) = p(N, T_N) = i \Rightarrow pattern(M) = pattern(N) \Rightarrow M \cong N$
2. $pP_M = pP_N = K \qquad K \in \mathbf{Keys}$
   $\Rightarrow$
   $p(M, T_M) = p(N, T_N) = K \Rightarrow pattern(M) = pattern(N) \Rightarrow M \cong N$
3. $pP_M = pP_N = P._p \qquad P._p \in \mathbf{pExp}$
   $\Rightarrow$
   $p(M, T_M) = p(N, T_N) = \otimes \Rightarrow pattern(M) = pattern(N) \Rightarrow M \cong N$
4. $pP_M = \{pP_M(L, T_M)\}_K = \{pP_N(L', T_N)\}_K = pP_N \Rightarrow$
   $pP_M(L, T_M) = pP_N(L', T_N)$
   $\Rightarrow$ *by induction hypothesis*
   $p(L, T_M) = p(L', T_N) \Rightarrow p(M, T_M) = \{p(L, T_M)\}_K = \{p(L', T_N)\}_K = p(N, T_N) \Rightarrow pattern(M) = pattern(N) \Rightarrow M \cong N$
5. $pP_M = (pP_M(L_1, T_M), pP_M(L_2, T_M)) = (pP_N(L'_1, T_N), pP_N(L'_2, T_N)) = pP_N$
   $\Rightarrow$
   $pP_M(L_1, T_M) = pP_N(L'_1, T_N) \ \wedge \ pP_M(L_2, T_M) = pP_N(L'_2, T_N)$
   $\Rightarrow$ *by induction hypothesis*
   $p(L_1, T_M) = p(L'_1, T_N) \ \wedge \ p(L_2, T_M) = p(L'_2, T_N) \Rightarrow$
   $p(M, T_M) = (p(L_1, T_M), p(L_2, T_M)) = (p(L'_1, T_N), p(L'_2, T_N)) = p(N, T_N) \Rightarrow$
   $pattern(M) = pattern(N) \Rightarrow M \cong N$ ■

5 Approximating Probabilistic Equivalence

The notion of probabilistic equivalence given above is extremely strict. In practice, it could be very difficult to find blocks that can be decrypted exactly with the same probability. As a consequence, two expressions that contain the same information and that yield probabilistic patterns where corresponding pieces of information can be obtained with similar probability (but not exactly the same) would not be probabilistically equivalent.

In this section, we introduce a compatibility relation, called *$\varepsilon-$probabilistic similarity* ($\approx_\varepsilon$), which approximates the probabilistic equivalence by taking into account small fluctuations of the probabilistic parameters associated to the probabilistic expressions.

So, we say that the expressions M and N are $\varepsilon-$probabilistically similar ($M \approx_\varepsilon N$) if they yield the same probabilistic pattern (except for small fluctuations) and if $pMax_M$ and $pMax_N$ are “almost the same”. Formally, we have:

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

In the definition above, we introduced a new relation ($=_\varepsilon$) for probabilistic patterns, formally defined by the following rules:

$K =_\varepsilon K$		$K \in \mathbf{Keys}$
$i =_\varepsilon i$		$i \in \mathbf{Bool}$
$(P, Q) =_\varepsilon (P', Q')$	$\Leftrightarrow P =_\varepsilon P' \ \wedge \ Q =_\varepsilon Q'$	$P, Q, P', Q' \in \mathbf{pPat}$
$\{P\}_K =_\varepsilon \{P'\}_K$	$\Leftrightarrow P =_\varepsilon P'$	$P, P' \in \mathbf{pPat},\ K \in \mathbf{Keys}$
$P_{\cdot p} =_\varepsilon P'_{\cdot p'}$	$\Leftrightarrow P_{\cdot p} \sim_\varepsilon P'_{\cdot p'}$	$P_{\cdot p}, P'_{\cdot p'} \in \mathbf{pExp}$

The compatibility relation $\sim_\varepsilon$ introduced above is defined by the following rules:

$K_{\cdot p} \sim_\varepsilon K_{\cdot p'}$	$\Leftrightarrow \lvert p - p'\rvert \leq \varepsilon$	$K \in \mathbf{Keys}$
$i_{\cdot p} \sim_\varepsilon i_{\cdot p'}$	$\Leftrightarrow \lvert p - p'\rvert \leq \varepsilon$	$i \in \mathbf{Bool}$
$(P_{\cdot p_1}, Q_{\cdot p_2})_{\cdot p_3} \sim_\varepsilon (P'_{\cdot p'_1}, Q'_{\cdot p'_2})_{\cdot p'_3}$	$\Leftrightarrow \lvert p_3 - p'_3\rvert \leq \varepsilon$	$\wedge$
	$P_{\cdot p_1} \sim_\varepsilon P'_{\cdot p'_1} \ \wedge \ Q_{\cdot p_2} \sim_\varepsilon Q'_{\cdot p'_2}$	
	$P_{\cdot p_1}, Q_{\cdot p_2}, P'_{\cdot p'_1}, Q'_{\cdot p'_2} \in \mathbf{pExp}$	

Example 9. Consider the following expressions:

$$M = \{0\}_K \qquad N = \{0\}_{K'}.$$

If we assume $p_1 = p_{dec}(\{0\}_K)$ and $p_2 = p_{dec}(\{0\}_{K'})$, we have:

$$\begin{array}{lcl} pP_M = 0_{\cdot p_1} & \text{and} & pP_N = 0_{\cdot p_2}, \\ pMax_M = p_1 & \text{and} & pMax_N = p_2. \end{array}$$

If p_1 and p_2 are very similar values, 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 $M \not\approx N$.

It is worth noting that in case $\varepsilon = 0$ the $\varepsilon-$probabilistic similarity reduces to the probabilistic equivalence. It is also easy to see that if two expressions M and N are $\varepsilon-$probabilistically similar ($M \approx_\varepsilon N$) then M and N are equivalent ($M \cong N$). The proof is a trivial application of that of Theorem 1. Intuitively, with respect to the probabilistic equivalence, the notion of $\varepsilon-$probabilistic similarity expresses a weaker condition on the probabilistic information only, but not on the form of the expressions. Hence, as a consequence, we have the following result.

Lemma 1. *Given $M, N \in \mathbf{Exp}$, if $\exists \varepsilon \in [0, 1[: M \approx_\varepsilon N$, then $M \cong N$.*

6 Conclusion

The treatment of cryptographic operations within formal models is covered by a large and well-established body of literature, but most of these efforts do not consider cryptographic operations in an imperfect cryptography scenario. With this work, we start to fill this gap in order to offer the means for defining a formal cryptographic language where: (i) information leakage due to cryptanalysis can be estimated by employing $\approx$ and conditional statements, (ii) probabilistic covert channels can be studied by verifying non-interference security properties (as done, e.g., in $[5, 2]$).

In this setting, the similarity relation $\approx_\varepsilon$ can be used, in combination with an approximated definition of non-interference, to verify whether the privacy of cryptographic protocols can be guaranteed at a reasonable level. Finally, we point out that $\approx_\varepsilon$ (which is coarser than $\approx$) is also useful to partially mitigate the effect of the use of function $p_{dec}(\{N\}_K, G, \mathcal{A})$, which probabilistically estimates the information leakage by abstracting away from the computational model. However, as a future work, we plan to give a computational definition of such a function.

References

1. M. Abadi, P. Rogaway, *"Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)"*, Proc. of 1st IFIP Int. Conf. on Theoretical Computer Science, Springer LNCS 1872:3-22, 2000.
2. A. Aldini, M. Bravetti, R. Gorrieri, *"A Process-algebraic Approach for the Analysis of Probabilistic Non-interference"*, Journal of Computer Security, to appear.
3. M. Backes, B. Pfitzmann, *"Computational Probabilistic Non-interference"*, Proc. of 7th Europ. Symp. on Research in Comp. Sec., Springer LNCS 2502:1-23, 2002.
4. R. A. DeMillo, N. A. Lynch, M. Merritt, *"Cryptographic Protocols"*, Proc. of 14th Annual ACM Symposium on Theory of Computing, 1982.
5. A. Di Pierro, C. Hankin, H. Wiklicky, *"Approximate Non-Interference"*, Proc. of 15th Computer Security Foundations Workshop, IEEE CS Press, pp. 1-17, 2002.
6. D. Dolev, A. Yao, *"On the Security of Public-key Protocols"*, IEEE Transactions on Information Theory 29:198-208, 1983.
7. J. W. Gray III, *"Toward a Mathematical Foundation for Information Flow Security"*, Journal of Computer Security 1:255-294, 1992.
8. R. A. Kemmerer, *"Analyzing Encryption Protocols using Formal Verification Techniques"*, IEEE J. on Selected Areas in Communications, 7(4):448-457, 1989.
9. P. Lincoln, J. C. Mitchell, M. Mitchell, A. Scedrov *"A Probabilistic Poly-Time Framework for Protocol Analysis"*, Proc. of 5th ACM Conf. on Computer and Communications Security, ACM Press, pp. 112-121, 1998.
10. 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.
11. L. C. Paulson, *"The Inductive Approach to Verifying Cryptographic Protocols"*, Journal of Computer Security, 6(1-2):85-128, 1998.
12. S. Schneider, *"Security Properties and CSP"*, IEEE Symposium on Security and Privacy", pp. 174-187, 1996.