Completeness Theorems for the Abadi-Rogaway Language of Encrypted Expressions*

Daniele Micciancio Bogdan Warinschi
Department of Computer Science and Engineering,
University of California, San Diego
9500 Gilman Drive, CA 92093
{daniele,bogdan}@cs.ucsd.edu

Abstract

We show that the Abadi-Rogaway logic of indistinguishability for cryptographic expressions is not complete by giving a natural example of a secure encryption function and a pair of expressions, such that the distributions associated to the two expressions are computationally indistinguishable, but equality cannot be proved within the logic. We then introduce a new property for encryption schemes, called confusion freeness, and show that the Abadi-Rogaway logic (as well as an extension of it due to Jürjens) is sound and complete, whenever the encryption scheme that is used is confusion free. We prove confusion freeness to be a consequence of a natural security property, that of authenticated encryption. In addition, we consider a refinement of the logic that overcomes certain limitations of the original proposal, allowing for encryption functions that do not hide the length of the message being sent. Both the soundness theorem of Abadi and Rogaway, and our completeness result for authenticated encryption easily extend to this more realistic notion of secrecy.

1 Introduction

In the last two decades, the logic and computational complexity communities have developed two related, though quite different, approaches to the theoretical modeling of security. Each approach has its own benefits and shortcomings. On the one hand, the logic approach, usually based on some formal language, is primarily intended to make the task of writing and verifying cryptographic protocols more manageable. However, this approach is often criticized by cryptographers because it is not clear how well the semantics of the language captures the right intuition of cryptographic security.[1] On the other hand, formalizing and clarifying the meaning of common security notions like *secrecy*, *authenticity*, etc., has been one of the primary goals and major achievements of the complexity-based approach to cryptography. The stronger and more comprehensive adversarial model used by this approach has gained wide acceptance in the cryptographic community as the one providing the "right" notion of security. However, the relatively unstructured way in which security is formalized in this framework makes proving the security of even moderately complex protocols quite a formidable task.

Recently, several attempts have been made to bridge the gap between these two worlds [13, 9, 12]. Still, most of these approaches either do not retain the simplicity of the original logic formulations, or do not capture the complexity-theoretic notion of security in its full power. In this context, the logic introduced by Abadi and Rogaway in [2] deserves special attention. The logic allows reasoning about the elements of a simple language (**Exp**) of expressions that are built from basic messages and cryptographic keys using the pairing and encryption operations. For example, $E = ((\{K_1\}_{K_4}, \{K_3\}_{K_2}), K_2)$ is an expression in **Exp**, where $\{M\}_K$ denotes the encryption of M under the key K. Following the style of one community or the other, the elements of **Exp** are assigned two different interpretations. On the one hand, each expression has

*Research supported in part by NSF Career Award CCR-0093029

[1] In fact, many specific formalisms have been subject to much debate already within the formal methods community, and there does not even seem to be a consensus yet about which specific formalism best captures the desired security notions.

associated a *pattern* that intuitively captures the information an eavesdropper can legitimately obtain from seeing the expression, without prior knowledge of any key. For example, the pattern of E is $((\Box, \{K_3\}_{K_2}), K_2)$, where $\Box$ denotes an indecipherable string. On the other hand, each expression has a naturally associated a probability distribution over bit-strings, where the strings are obtained applying the (key generation and encryption) algorithms corresponding to a concrete cryptosystem. The main result of [2] is to show that under certain (standard) security requirements on the cryptosystem, if two expressions have the same pattern, then their naturally associated distributions are computationally indistinguishable. In other words, if equivalence between two expressions can be proved within the logic, then the corresponding distributions are guaranteed to be equivalent also under their standard computational interpretation.

The cryptographic language of [2] is clearly limited since it allows modeling only the simplest protocols where one party sends a sequence of (complex) messages to another one. However, it is a nice example of a simple language with a clean semantics that is sound according to a standard notion of security of complexity-based cryptography. This makes the result of [2] an important step towards languages that can be used to model more complex cryptographic protocols. In [1], the basic framework of [2] is extended to more complex protocols, where several parties interact by encrypting, decrypting and sending messages. The techniques used in [1] are similar: the execution of the protocol is represented abstractly by a set of patterns, each corresponding to a message sent or received by a party. The computational task faced by the adversary is the same: given the actual messages that are sent and received during the execution of the protocol, try to gain additional knowledge, beside the legitimate information revealed by the patterns. Notice that, as in [2], the adversary acts as a passive observer, i.e., the adversary cannot alter the control flow of the protocol, change messages, or inject messages in the communication network.

In this paper we take a closer look at the semantics of [2, 1] and consider the following completeness issue: if the probability distributions associated to two expressions are indistinguishable, is it true that the two expressions have the same pattern? In other words, we ask whether the pattern associated to an expression fully characterizes the information recoverable by an adversary, or it is possible for two expressions to be equivalent in the computational setting, and still, their equivalence cannot be proved using the Abadi-Rogaway pattern semantics. For simplicity we focus on the basic framework of [2], and then we show that our results can be easily extended to arbitrary protocols as considered in [1] (See Section 7).

In [2], Abadi and Rogaway give an example to illustrate that their semantics is not complete. However, the example relies on a pathological situation which can be easily avoided. More specifically, [2] considers encryption functions that hide all information about the input, including its length. It is easy to see that this strong notion of secrecy can be achieved only if an a priori bound is put on the length of the input. If this is the case, the encryption of two messages that have different patterns, but lie outside this restricted message space, lead to the same probability distribution ("invalid message"). If we consider general purpose encryption schemes, i.e. schemes that can be used to encrypt arbitrary messages, (or we require the encryption function to be applied only to messages in the prescribed range,) then the example of [2] breaks down, leaving the question of completeness open.

In this paper we show that the Abadi-Rogaway semantics is not complete, even for general purpose encryption schemes, under the constraint that all messages to be encrypted belong to the message space. Then, we consider whether a completeness theorem can be proved under the assumption that the encryption function satisfies a stronger notion of security than the one considered in [2]. Namely, we consider *authenticated encryption*, i.e., encryption schemes providing both secrecy and message integrity. This is a stronger, yet natural and relatively standard, notion of security and it has been the subject of many recent research papers in the cryptographic community [5, 3, 11]. Interestingly, we can show that if authenticated encryption is used, then the Abadi-Rogaway pattern semantics is both sound and complete. In fact, it is enough to assume that the encryption scheme satisfies a "confusion-freeness" property (independently defined in [1], and implied by the definition of authenticated encryption) which informally states that attempting to decrypt a ciphertext with a randomly and independently chosen key will fail with all but negligible probability. Following a suggestion of [2], we extend the pattern semantics to deal with encryption schemes that do not hide the length of the message, and observe that both the soundness and completeness results hold for these more realistic encryption schemes as well. The kind of technique we use to prove our completeness result is quite general, and we illustrate this by sketching an extension of our result to the more complex logic of [1].

The rest of the paper is organized as follows. In Section 2 we give some background about the computational approach to cryptography. In Section 3 we review the logic of encrypted expressions of [2]. The incompleteness of the logic is demonstrated in Section 4 by means of a more natural counterexample than the one considered in [2]. In Section 5, we prove our completeness result under the appropriate assumptions on the encryption scheme. This is the main technical contribution of this paper. In the following two sections we sketch two extensions of our results. In Section 6, we extend the pattern semantics of [2] to deal with encryption schemes that do not hide the length of the message, and observe that both the soundness and completeness results hold for these more realistic encryption schemes as well. Then, in Section 7 we show that our techniques can be easily extended to the more complex logic of [1], yielding a similar completeness result. Section 8 concludes with a discussion of other possible extensions of our results and open problems.

2 Preliminaries

In this section we recall some of the terminology and concepts used by the complexity based approach to cryptography. These notions are all relatively standard (except possibly some terminology from [2] about security levels for encryption) and this section can be safely skipped by the reader familiar with modern cryptography. For a through introduction to the subject the reader is referred to [7].

In the computational setting, the notion of efficient computation is identified with polynomial time computability. Formally, a function f (say, from bit strings to bit strings) is efficiently computable if there exists a Turing machine M and a polynomial p such that for any input x, $M(x)$ halts in at most $p(|x|)$ steps with output $f(x)$. In cryptographic applications it is interesting to consider randomized algorithms as well that succeed with some probability. Running times and probabilities are usually measured as function of a security parameter $\eta \in \mathsf{N}$, so that the cryptographic functions can be made arbitrarily hard to break by appropriately choosing the value of η. It is implicitly assumed that the size of all inputs to the algorithms is bounded by a polynomial function of the security parameter.

Definition 2.1 A function $\nu : \mathsf{N} \to [0, 1]$ is *negligible* if for any polynomial p there exists n_0 such that $\nu(n) \leq 1/p(n)$ for any $n_0 \leq n$.

The definition of negligible function naturally corresponds to the identification of efficient computation with polynomial time. In particular, if an experiment succeeds with negligible probability, then repeating the experiment any polynomial number of times will always fail, except with negligible probability. We say that an event has overwhelming probability, if is happens with probability $1 - \nu(\eta)$ for some negligible function ν. A probability ensemble is a sequence of probability distributions $\{D_\eta\}_{\eta \in \mathsf{N}}$, indexed by the security parameter η. Usually D_η is a probability distribution over the set $\{0, 1\}^{\mathrm{poly}(\eta)}$ of binary strings of length polynomial in the security parameter.) A notion that plays a fundamental role in the complexity based approach to cryptography is that of computational indistinguishability.

Definition 2.2 Two probability ensembles $\{A_\eta\}_\eta$ and $\{B_\eta\}_\eta$ are computationally indistinguishable if for any probabilistic polynomial time computable predicate D,

$$| \Pr_{x \in A_\eta} [D(x)] - \Pr_{x \in B_\eta} [D(x)]|$$

is a negligible function of η.

An encryption scheme is a triple of (possibly probabilistic) algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where

- $\mathcal{K}$ (the *key generation algorithm*) on input a security parameter η, outputs a randomly chosen key k from some domain Key.
- $\mathcal{E}$ (the *encryption algorithm*) takes as input a key k and a *plaintext* m, and outputs a *ciphertext* $c = \mathcal{E}_k(m)$.
- $\mathcal{D}$ (the *decryption algorithm*) takes as input a key k and a string c and outputs the underlying plaintext $p = \mathcal{D}_k(c)$ if the decryption succeeds, or $\perp$ otherwise.

Any encryption scheme should satisfy the standard correctness requirement that for any key k and message m, $\mathcal{D}_k(\mathcal{E}_k(m)) = m$. Informally, an encryption scheme is secure [8] if $\mathcal{E}_k(m)$ hides all partial information about the message m, except possibly its length. This is a much stronger requirement than simply asking that computing m from $\mathcal{E}_k(m)$ should be hard, and it is important in many practical scenarios, where, for example, the message m comes from a small message space (e.g., $\{\text{yes}, \text{no}\}$), or the adversary has some a-priori knowledge about the message (e.g., the message is something of the form "the password of the day is ..."). The reason why encryption schemes are not usually required to hide the length of the message is that arbitrarily long messages requires arbitrarily long ciphertexts to be represented. So, the only way to hide the length of a message, is to put an absolute bound on the length of any allowable message (or equivalently, restricting the message space to a finite set) and use ciphertexts of the same length for all the messages. The intuition that a good encryption scheme should hide all partial information about the messages (except possibly their length), can be captured requiring that for any two messages m_0, m_1 (of the same length), distribution ensembles $\{\mathcal{E}_k(m_0)\}_\eta$ and $\{\mathcal{E}_k(m_1)\}_\eta$ are computationally indistinguishable. For each security parameter η, the probability distributions are defined over the choice of the key k, and the randomness used by the encryption algorithm. Using the fact that computational indistinguishability is an equivalence relation, one can equivalently require that for any message m, the ensemble $\{\mathcal{E}_k(m)\}_\eta$ should be computationally indistinguishable from $\{\mathcal{E}_k(0_{|m|})\}_\eta$, where $0_{|m|}$ denotes some standard message (e.g., the all zeros message) of the same length as m. Picturesquely, this definition is usually described in terms of an *adversary* $\mathcal{A}$ trying to "break" the scheme: first, a key k is chosen uniformly at random running a key generation algorithm $\mathcal{K}$; then, the adversary $\mathcal{A}$, not knowing the key k, chooses a message m. Having the adversary choose m corresponds to the universal quantification over message m above. finally, $\mathcal{A}$ is given a ciphertext drawn at random either from distribution $\mathcal{E}_k(m)$ or $\mathcal{E}_k(0_{|m|})$. The adversary is considered successful if it correctly guesses which distribution was used to select the ciphertext. (Say, A should output 1 if $c = \mathcal{E}_k(m)$, and 0 if $c = \mathcal{E}_k(0_{|m|})$.) Clearly, the adversary can always succeeds with probability 1/2 by outputting a uniformly random guess in $\{0, 1\}$. The advantage of the adversary $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ succeeds minus 1/2. Replacing the probability that $\mathcal{A}$ outputs 0 on input an encryption of $0_{|m|}$ with one minus the probability of the opposite event, simple calculations show that the advantage of $\mathcal{A}$ is also equal to

$$\frac{1}{2}\left(\Pr\{\mathcal{A}(\mathcal{E}_k(m)) = 1\} - \Pr\{\mathcal{A}(\mathcal{E}_k(0_{|m|})) = 1\}\right).$$

So, the adversary's advantage is negligible if and only if distributions $\mathcal{E}_k(m)$ and $\mathcal{E}_k(0_{|m|})$ are computationally indistinguishable. It can be shown using standard techniques (see [8]) that allowing the adversary to submit polynomially many messages (and receive either their encryptions, or the encryption of $0_{|m|}$) instead of a single one, does not make the definition any stronger. Formally, we consider an adversary $\mathcal{A}$ that is given oracle access to one of two possible encryption procedures: the first one $\mathcal{E}_k(\cdot)$ on input m outputs an (independently and randomly chosen) encryption of the message m, the second one $\mathcal{E}_k(0_{|m|})$ replaces the input m with a message of the same length as m consisting of all zeros, and returns an encryption of this message. We denote by $\mathcal{A}^{\mathcal{O}}$ the result of running $\mathcal{A}$ with oracle access to $\mathcal{O}$. The encryption scheme is secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage

$$\Pr[A^{\mathcal{E}_k(\cdot)}(\eta) = 1] - \Pr[A^{\mathcal{E}_k(0_{|\cdot|})}(\eta) = 1]$$

is a negligible function of security parameter η. The probabilities are computed over the random choice of key k, and the randomness used by the adversary and the encryption oracles. This is the basic notion of security introduced in [8], and known (in this or other equivalent formulations) as "indistinguishability" or "semantic security" under chosen plaintext attack. "Chosen plaintext attack" refers to the fact that the adversary is allowed to see ciphertexts of messages of its choice.

Abadi and Rogaway [2] consider many possible variants of the definition described above. Specifically, they consider encryption schemes that hide the length of the message (for some fixed finite message space), or hide whether different ciphertexts were obtained using the same or different keys. These stronger notions are captured by modifying the experiment above as follows. For encryption schemes that hide the length of the message, oracle $\mathcal{E}_k(0_{|\cdot|})$ is replaced by an oracle that ignores the input message completely, and outputs the encryption of a standard message $\mathcal{E}_k(0)$ of some fixed length. Hiding whether the same key is being used to encrypt multiple messages is captured by replacing oracle $\mathcal{E}_k(\cdot)$ with a pair of oracles $\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)$ that encrypt their input according to two independently chosen keys k, k', while oracle $\mathcal{E}_k(0)$ is replaced by two identical oracles $\mathcal{E}_k(0), \mathcal{E}_k(0)$ that encrypt standard message 0 with respect to the same random key k (but possibly using different randomness of course). All possible combinations of these three security requirements are considered in [2] and labeled with the numbers from 0 to 7, with "type-0" security corresponding to a scheme that hides all information, including the length of the message and the key used to encrypt. This is the notion of security [2] focuses on. For completeness, we repeat the formal definition below.

Definition 2.3 [Type 0 secure encryption] An encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is *type-0* secure if for any polynomial time algorithm A (with access to two encryption oracles) the quantity

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

is a negligible function of η.

3 The Abadi-Rogaway Logic

In this section we recall the main ideas behind the Abadi-Rogaway logic.

3.1 Syntax

The logic allows reasoning about expressions in the language **Exp** defined as follows. We consider a fixed finite set of *block symbols*, **Block** that represents some fixed data set. This is a slight generalization of the language of [2] where the basic data set is **Bool** and only contains symbols representing the bits $0, 1$. One could imagine for example, that **Block** contains symbols for each of the 2^l bit sequences of some fixed length l. We also consider a fixed set of key symbols, denoted **Keys**. We use $K, K', K'', ..., K_1, K_2, ...$ to denote elements of Key. The elements of **Exp** are built from **Keys** and **Block** by using the pairing and encryption operations. Formally, **Exp** is the language generated by the following grammar:

$$\begin{array}{rll} M ::= & K & \text{key (for K} \in \textbf{Keys}) \\ & B & \text{block (for B} \in \textbf{Block}) \\ & (M, M) & \text{pair} \\ & \{M\}_K & \text{encryption (for K} \in \textbf{Keys}) \end{array} \tag{1}$$

Two possible interpretations are associated to the elements of **Exp**. The first, called the formal semantics, is a pattern. The second, called computational semantics, is a distribution ensemble on bit strings.

3.2 Formal and computational semantics

FORMAL SEMANTICS. The formal semantics is defined in the spirit of the formal methods community, and it is purely syntactical. It associates to each expression $E \in \textbf{Exp}$ an element $\mathsf{pattern}(E) \in \textbf{Pattern}$ from the language generated by the following grammar which extends (1) with a special symbol $\Box$ used to denote indecipherable messages:

$$\begin{array}{rll} P ::= & K & \text{key (for K} \in \textbf{Keys}) \\ & B & \text{block (for B} \in \textbf{Block}) \\ & (P, P) & \text{pair} \\ & \{P\}_K & \text{encryption(for K} \in \textbf{Keys}) \\ & \Box & \text{undecryptable} \end{array} \tag{2}$$

Computing the pattern attached to some expression E is done in two steps. The first step is to compute a set of keys, denoted $\mathsf{recoverable}(E)$. Intuitively, this set contains all keys that can be computed by an adversary after seeing E. It contains all keys that are sent in the clear, those keys that can be obtained decrypting messages with keys sent in clear and so on. Different formal definitions are possible for the function $\mathsf{recoverable}(\cdot)$. In the sequel we present an iterative formulation that will be useful when proving our completeness result. The second step is to use the keys in $\mathsf{recoverable}(E)$ to decrypt as many subexpressions of E as possible.

We start by giving a recursive definition for a *key recovery function* $\mathsf{F_{kr}}$ that on input $(E,T) \in \mathbf{Exp} \times \mathcal{P}(\mathbf{Keys})$, returns the set of keys recoverable from E using for decryption *only* the keys in T. The definition, given in Figure 1, is the obvious one. For example, Equation (5) in Figure 1 says that if key K is in T, then the set of keys that can be recovered from $\{E\}_K$ using the keys in T, is equal to the set of keys that can be recovered directly from E (using the same set of keys). Similarly, Equation (4) in Figure 1 says that if K is not in T, then no other keys are recoverable from $\{E\}_K$, beside those already known.

Next, we define $\mathsf{recoverable}(E)$, the set of keys recoverable from expression E, as the least fixed point of the function $\mathsf{F_{kr}}(E,\cdot)$, with respect to the set inclusion ordering relation. Algorithmically, this set can be easily obtained as follows. Given expression E, we inductively define the sets $G_0(E), G_1(E), G_2(E), \ldots$, where $G_i(E)$ is the set of keys that can be recovered from E using (at most) i applications of the key recovery function, starting with no keys. Then, $\mathsf{recoverable}(E) = \bigcup_i G_i(E)$. Notice that $\emptyset = G_0(E) \subseteq G_1(E) \subseteq \ldots \subseteq G_i(E) \subseteq \ldots$ and all $G_i(E)$ are subsets of key symbols appearing in E. Therefore, for all sufficiently large i the equality $G_i(E) = G_{i+1}(E)$ holds. If i is an upper bound on the number of distinct key symbols appearing in E (e.g., $i = |E|$,where $|E|$ is the size of expression E) then $\mathsf{recoverable}(E) = G_i(E) = G_{|E|}(E)$.

1. $\mathsf{F_{kr}}(B,T) = T$;
2. $\mathsf{F_{kr}}(K,T) = \{K\} \cup T$;
3. $\mathsf{F_{kr}}((E_0,E_1),T) = \mathsf{F_{kr}}(E_0,T) \cup \mathsf{F_{kr}}(E_1,T)$;
4. $\mathsf{F_{kr}}(\{E\}_K,T) = T$, if $K \notin T$;
5. $\mathsf{F_{kr}}(\{E\}_K,T) = \mathsf{F_{kr}}(E,T)$, if $K \in T$;

- $G_0(E) = \emptyset$;
- $G_i(E) = \mathsf{F_{kr}}(E, G_{i-1}(E))$
- $\mathsf{recoverable}(E) = \bigcup_i G_i(E) = G_{|E|}(E)$

Figure 1: Definition of the set of keys recoverable from E

Example. If $E = (\{\{K_2\}_{K_1}\}_{K_1}, \{K_3\}_{K_2}, K_1)$ then $G_1(E) = \mathsf{F_{kr}}(E,\emptyset) = \{K_1\}$ and $G_2(E) = \mathsf{F_{kr}}(E,\{K_1\}) = \{K_1,K_2\}$ and $\mathsf{recoverable}(E) = G_3(E) = G_4(E) = \{K_1,K_2,K_3\}$.

Assume the adversary is in possession of a set of keys $T \subseteq \mathbf{Keys}$. Then, the adversary's view of an expression $E \in \mathbf{Exp}$ can be described as a pattern $\mathsf{p}(E,T) \in \mathbf{Pattern}$ defined inductively by:

$$\begin{aligned}
&\mathsf{p}(K,T) = K \\
&\mathsf{p}(B,T) = B \\
&\mathsf{p}((M,N),T) = (\mathsf{p}(M,T), \mathsf{p}(N,T)) \\
&\mathsf{p}(\{M\}_K,T) = \{\mathsf{p}(M,T)\}_K && \text{(if } K \in T) \\
&\mathsf{p}(\{M\}_K,T) = \Box && \text{(if } K \notin T)
\end{aligned}$$

where $K \in \mathbf{Keys}, B \in \mathbf{Block}$ and $M,N \in \mathbf{Exp}$. The pattern associated to an expression is naturally defined as:

$$\mathsf{pattern}(E) = \mathsf{p}(E, \mathsf{recoverable}(E))$$

Two expressions $M,N \in \mathbf{Exp}$ are equivalent, denoted $M \equiv N$, if $\mathsf{pattern}(M) = \mathsf{pattern}(N)$. Note that this equivalence does not identify expressions such as $(\{0\}_{K_1}, K_1)$ and $(\{0\}_{K_2}, K_2)$, which only differ by the names used to represent the keys. The fact that key names are not significant is captured by the *equivalence up to renaming* relation, denoted $\cong$, and defined as follows:

$$M \cong N \text{ if and only if there exists a injection } \sigma : \mathbf{Keys} \to \mathbf{Keys} \text{ such that } M \equiv N\sigma$$

where $N\sigma$ is the result of applying σ to the keys in N.

Computational semantics. For any fixed encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$, and security parameter $\eta \in N$ (see Section 2), an expression $E \in \mathbf{Exp}$ defines a natural probability distribution on strings. This distribution, denoted $[\![E]\!]_{\Pi(\eta)}$, is obtained as follows. The key symbols that appear in the expression are assigned key values using the key generation algorithm $\mathcal{K}$. (The same key value is assigned to identical key symbols, and different symbols are assigned using independent executions of $\mathcal{K}$.) We denote by $\tau(K)$ the key value associated to $K \in \mathbf{Keys}$, and we refer to τ as *the key assignment underlying* $[\![E]\!]_{\Pi(\eta)}$. The rest of the definition is recursive. Each basic block $B \in \textbf{Block}$ is mapped to a fixed bit string, the bit string x corresponding to a pair $E = (E_0, E_1)$ is obtained from the bit strings corresponding to E_0 and E_1 respectively, the image of a formal encryption $\{E\}_K$ is obtained by running the encryption algorithm on the bit string x recursively associated to E using key $\tau(K)$ and independent uniformly random coin tosses. The specific way expressions are encoded into bit strings is application dependent and not particularly important. The only relevant property is that the encoding function is unambiguous, i.e., the type of the expression is univocally determined by the bit string and the context in which it occurs, and the bit strings corresponding to the component subexpressions can also be unambiguously recovered. For example, [2] considers a representation where expressions are mapped to bit strings consisting of the encoding of subexpressions, followed by a type tag representing one of the basic types {"*block*", "*key*", "*pair*", "*ciphertext*"}. Special delimiter symbols are used when appropriate, e.g., pairs are encoded using special delimiters to mark the end of the first subexpression and the beginning of the second one. Alternatively, one can use a prefix free encoding of the subexpressions, followed by the type tag, making the use of delimiter symbols unnecessary.

We remark that this unambiguity property does not play a fundamental role in the soundness result of [2], but it is *essential* for achieving *our* completeness result. Nevertheless, the specific way the property is achieved is not important. For example, the type of an expression could be inferred from the context in which the expression occurs, instead of being explicitly included as a type tag in the actual bit representation. Consider for example a protocol where party A sends to party B a (fixed length) data block N (identifying the protocol type) followed by the encryption $E_K(M, M)$ of a pair, where the actual message is repeated twice (e.g., as a sort of redundancy check). In this case, no type tags or delimiters of sorts are required: party B simply reads the first data block N identifying the "repeat and encrypt" protocol, applies the decryption algorithm (using a key possibly specified also in the first data block) to the remaining portion of the message to recover (M, M). At this point B checks that the cleartext has even length, and that the first half of it equals the second part.

In the rest of the paper, we assume that the type and arguments of an expression can always be recovered by its bit string representation, and write $\langle x, tag\rangle$ to denote the representation of an expression of type tag, whether or not tag is explicitly included in the actual bit string.

The *computational semantics* of E is the distribution ensemble $([\![E]\!]_{\Pi(\eta)})_\eta$ (i.e. a distribution $[\![E]\!]_{\Pi(\eta)}$ on strings for each security parameter η). Two expressions are equivalent (under the computational semantics) if their associated distribution ensembles are computationally indistinguishable (as defined in Section 2).

3.3 The soundness result

The main result of [2] is a relation between the formal and the computational semantics described above. The relation holds for *acyclic* expressions and encryption schemes that are type-0 secure (see Section 2 for the definition of type-0 security).

We recall the definition of acyclicity. We say that K encrypts K' in M, if there exists an expression N such that $\{N\}_K$ is a subexpression of M and K' occurs in N (except possibly as an encryption key). This defines a binary relation on keys, the "encrypts" relation. We say that M is an acyclic expression if its associated "encrypts" relation does not contain any cycles, i.e., a sequence of keys $K_0, \ldots, K_l$ such that K_{i-1} encrypts K_i for all $i = 1 \ldots, l$ and $K_0 = K_l$. For example $E_{K_1}(E_{K_3}(E_{K_2}(A)), E_{K_3}(K_2))$ and $E_{K_1}(E_{K_3}(E_{K_1}(A)), E_{K_3}(K_2))$ are acyclic, while $E_{K_1}(E_{K_2}(K_3), E_{K_3}(K_1))$ is not because it contains a cycle (K_1, K_3, K_1). The use of encryption functions in cyclic expressions (also referred to "circular" use of encryption) is typically regarded as a bad security practice within the cryptographic community, as the standard cryptographic definition of security for encryption schemes requires the key to be chosen randomly and independently from the message being encrypted. The soundness theorem of [2] is the following.

Theorem 3.1 [Abadi-Rogaway] Let E_0 and E_1 be acyclic expressions and let Π be a type-0 secure encryption scheme. Suppose that $E_0 \cong E_1$, then $[\![E_0]\!]_{\Pi(\eta)} \approx [\![E_1]\!]_{\Pi(\eta)}$.

4 The logic is incomplete for general encryption

An important question, that is only briefly addressed in [2], regards the exact power of the formal approach. Is it the case that if indistinguishable ensembles are associated to different expressions, then the expressions can be proved equivalent within the logic? In other words, is the converse of Theorem 3.1 true? The authors of [2] provide a very simple counterexample. Consider an encryption scheme with limited plaintext space and two expressions representing the encryption (under a known key) of some messages for which the bit representation lies outside the plaintext space. Since the messages are different, the two expressions are mapped to different patterns. Still, since both messages cannot be encrypted, running the encryption algorithm will produce the same distribution: some error message specifying that the input is out of range.

In this section we provide a natural counterexample that avoids the pathological situation described above. We exhibit a secure encryption scheme and two expressions with different patterns, but with the same computational interpretation, and where the inputs to the encryption function always belong to the prescribed message space. We consider bit string representations both with and without explicit type tags. In each case, we define a specific encryption scheme (with possibly limited message space) which satisfies the standard notion of security considered in [2], and exhibits a corresponding counterexample to demonstrate that the logic of [2] is not complete. In both cases, our examples never involve the application of the encryption function on messages outside the prescribed range.

Untagged representation. Let $\{F_i\}_{i \in I_\eta}$ be a pseudorandom function family[2] , $F_i : \{0,1\}^\eta \to \{0,1\}^\eta$ and $I_\eta = \{0,1\}^\eta$. We define an encryption scheme $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ with plaintext space $\mathsf{P} = \{0,1\}^\eta$ as follows.

- the key generation algorithm $\mathcal{K}'$ on input η, picks uniformly at random an element of I_η (i.e. it selects a random function from the family);
- the encryption of $m \in \mathsf{P}$ under the key $k \in I_\eta$ is $\mathcal{E}'_k(m) = r||(F_k(r) \oplus m)$, where r is some freshly generated randomness, $||$ denotes string concatenation and $\oplus$ denotes bitwise exclusive or operation; to simplify notation, in the sequel we will omit the parenthesis, and assume that $\oplus$ has higher precedence than $||$.
- the plaintext corresponding to a ciphertext $(r||C)$ is recovered as $\mathcal{D}'_k(r||C) = F_k(r) \oplus C$;

The reader can easily verify that for any key k and message m, $\mathcal{D}_k(\mathcal{E}_k(m)) = m$. Moreover, the scheme is known to be type-0 secure, under the assumption that the function family $F_{i i}$ is pseudorandom.

Now consider expressions $E_1 = (\{K_3\}_{K_1}, K_1)$ and $E_2 = (\{K_3\}_{K_1}, K_2)$. The two distributions have patterns

$$\begin{aligned} \mathsf{pattern}(E_1) &= (\{K_3\}_{K_1}, K_1) \\ \mathsf{pattern}(E_2) &= (\square, K_2) \end{aligned}$$

which are not equivalent, even up to renaming. In other words, the two expressions are not equivalent according to the logic. Now consider the probability distributions associated to the two expressions for some security parameter η. These distributions are

$$(r_1||F_{k_1}(r_1) \oplus k_3, k_1) \text{ and } (r_2||F_{k_1}(r_2) \oplus k_3, k_2)$$

where r_1, r_2, k_1, k_2, k_3 are uniformly and independently distributed in $\{0,1\}^\eta$. We claim that the two distributions are identical. The bit string associated to the first expression consists of a random r_1 concatenated with $F_{k_1}(r_1) \oplus k_3$ followed by key k_1. The initial randomness r_1 and the trailing key k_1 are independent and uniformly random. Moreover, for any given r_1 and k_1, also the middle expression $F_{k_1}(r_1) \oplus k_3$ is distributed uniformly at random because it is the exclusive or of a fixed value $F_{k_1}(r_1)$ with an independent uniformly distributed key k_3. A similar reasoning shows that also the probability distribution associated to the second expression is completely random. In particular, the two distributions are identical, despite the fact that the two expressions are not equivalent according to the logic.

The discrepancy between the formal and computational semantics can be informally explained as follows. In the formal semantics it is implicitly assumed that decrypting $\{M\}_K$ succeeds only if the decryption key

[2]Informally, a family of function is pseudorandom, if a random element from the family look like a truly random function to any computationally bounded observer. The formal definition considers an adversary A that given oracle access to a function f has to decide if the function f belongs to the family. The probability of A accepting when function is chosen at random from the family should be the same (up to negligible terms) as when A is given access to a completely random function with the same domain and codomain as f.

is K, and that whoever performs the decryption is aware of the success/failure. A similar assumption does not hold in general for encryption schemes (such as the one we constructed above): holding a key k does not help in telling apart encryptions under k from encryptions under k' (for some $k' \neq k$). Note that if type tags are used, as is the case in [2], our counterexample breaks down: the image of a key symbol K is of the form $\langle \tau(K), \text{“}key\text{”}\rangle$ and decrypting an encryption of this plaintext with the wrong key can be detected by checking the tag. Nevertheless, our next construction shows that in principle, using type tags is not a solution to this problem, at least for encryption schemes with limited message space.

Tagged representations. Using the encryption scheme $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ defined above, we construct a new encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with plaintext space equal to $\mathsf{P} = \{\langle m, \text{“}key\text{”}\rangle | m \in \{0,1\}^\eta\}$, where η is a security parameter. The key generation algorithm $\mathcal{K}$ is the same as $\mathcal{K}'$. The encryption of a plaintext $\langle m, \text{“}key\text{”}\rangle \in \mathsf{P}$ under key k is $\mathcal{E}_k(\langle m, \text{“}key\text{”}\rangle) = \mathcal{E}'_k(m)$, i.e. the tag is ignored and the encryption function $\mathcal{E}'_k(\cdot)$ is applied. The decryption of a ciphertext c is $\mathcal{D}_k(c) = \langle \mathcal{D}'(c), \text{“}key\text{”}\rangle$, i.e. the ciphertext is decrypted with $\mathcal{D}'_k(\cdot)$ and the resulting plaintext is tagged with the tag “key” and output. It is easy to check that this is a correct, type-0 secure encryption scheme when only messages from P are encrypted. Now, consider the very two expressions used in the above counterexample. For security parameter η , their associated distributions are

$$(\langle r_1 || F_{k_1}(r_1) \oplus k_3, \text{“}ciphertext\text{”}\rangle, \langle k_1, \text{“}key\text{”}\rangle) \text{ and } (\langle r_2 || F_{k_1}(r_2) \oplus k_3, \text{“}ciphertext\text{”}\rangle), \langle k_2, \text{“}key\text{”}\rangle)$$

where r_1, r_2, k_1, k_2, k_3 are uniformly and independently distributed in $\{0,1\}^\eta$, which are again equal.

5 The logic is complete for authenticated encryption

In Section 4 we provided a counterexample that shows that type-0 security is not sufficient to achieve completeness for the Abadi-Rogaway logic. Moreover, the counterexample suggests that this is the case because, in general, it is not possible to determine whether a given ciphertext c was created with a given key k. We develop this idea, and show that when the encryption scheme Π satisfies a certain condition, called *confusion-freeness*, the Abadi-Rogaway logic is complete. We also show that confusion-freeness is implied by a standard security requirement on encryption schemes, namely *authenticity* [5, 3, 11, 14, 10, 6].

5.1 Confusion-free encryption and authenticated encryption

The confusion free property can be informally described as follows. It asks from an encryption scheme that whenever two keys are generated independently at random, decrypting with one key ciphertexts produced using the other key fails, except possibly with some negligible probability. We note that a similar notion of confusion-freeness has been independently defined in [1].

Definition 5.1 [Confusion freeness] Let $D = \{D_1(\eta), D_2(\eta), ..., D_l(\eta)\}$ a finite set of distributions ensembles and Π be a symmetric encryption scheme and indexed by a security parameter η. We say that Π is confusion free with respect to D iff there exists a negligible function ν_D such that for any $1 \leq i \leq l$

$$\Pr\left[k_1, k_2 \xleftarrow{R} \mathcal{K}(\eta), x \xleftarrow{R} D_i(\eta) : \mathcal{D}_{k_1}(\mathcal{E}_{k_2}(x)) \neq \bot\right] \leq \nu_D(\eta) \tag{3}$$

We say that Π is confusion free, if it is confusion free with respect to any finite set of distribution ensembles.

Remark 5.2 We could have equivalently defined confusion freeness with respect to a single distribution ensemble. We gave a definition involving a sequence of distributions in order to simplify the presentation of the proof of our completeness result. We anticipate that in our proofs D will be typically instantiated with $\{[\![E_1]\!]_{\Pi(\eta)}, [\![E_2]\!]_{\Pi(\eta)}, ..., [\![E_l]\!]_{\Pi(\eta)}\}$, where $E_1, E_2, ..., E_l$ are all subexpressions of some $E \in \mathbf{Exp}$, and η is the security parameter of the encryption scheme. In this case we will refer to ν_D as the function associated to expression E.

The concept of authenticated encryption has recently received quite a lot of attention ([5, 3, 11, 14, 6, 10]). Its fundamental goal, authenticity of communication, can be informally stated as follows: no polynomial time adversary, should be able to produce a valid ciphertext (a ciphertext that is not decrypted as $\bot$) even after seeing polynomially many encryptions of messages of his choice (except with negligible probability). While different flavors of authenticity are possible [5], in this paper we concentrate on a specific definition: integrity of plaintext, or INT-PTXT security, using the terminology of [5], defined below.

Definition 5.3 [Secure Authenticated Encryption Scheme] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. An adversary against the authenticity property of Π is a polynomial time algorithm A that has access to an encryption oracle $\mathcal{E}_k(\cdot)$ which allows A to see ciphertexts corresponding to plaintexts of its choice: when adversary A sends a query m to the oracle, the oracle answers with $\mathcal{E}_k(m)$, each time using independent randomness to encrypt the message. Security of an authenticated encryption scheme can be defined using the following experiment (defined for each fixed security parameter η): generate a random key k and instantiate the encryption oracle with $\mathcal{E}_k(\cdot)$. Next, we let the adversary A interact with the encryption oracle, until A stops and outputs an attempted forgery c. We say that the adversary "wins", if it produced a forgery, i.e., a ciphertext c such that $\mathcal{D}_k(c) \neq \bot$ (i.e., c is a valid ciphertext), and A never asked the encryption oracle to produce a ciphertext for $\mathcal{D}_k(c)$. An encryption scheme has the authenticity property if the probability that the adversary wins (taken over the coin tosses of the key generation algorithm and coin tosses of A) is a negligible function of the security parameter η.

The relation between secure authenticated encryption schemes and confusion freeness is captured by the following lemma.

Lemma 5.4 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a secure authenticated encryption scheme. Then Π is confusion free with respect to any finite set of polynomial time samplable distributions.

Proof: Let Π be a secure authenticated encryption scheme and $D = \{D_1(\eta), D_2(\eta), ..., D_l(\eta)\}$ be a set of polynomial time samplable distributions. For each $1 \leq i \leq l$ define an adversary A_i against the authenticity of Π that works as follows. A_i is given access to an encryption oracle $\mathcal{E}_k$ with an unknown, randomly chosen key k. However, our adversary A_i never invokes the oracle. It simply generates another random key k_i by running $\mathcal{K}$ with freshly generated coins, selects a random plaintext x according to distribution D_i and outputs $c = \mathcal{E}_{k_i}(x)$ as its attempted forgery. If c is a valid ciphertext with respect to key k (i.e., $\mathcal{D}_k(c) \neq \bot$), then c is also a forgery because A_i never queried the encryption oracle on any plaintext.[3] It follows from the security of Π, that the success probability of A_i

$$\nu_i(\eta) = \Pr\left[k \overset{R}{\leftarrow} \mathcal{K}(\eta), c \overset{R}{\leftarrow} A_i(\eta) : \mathcal{D}_k(c) \neq \bot)\right]$$

is a negligible function of security parameter η. Let $\nu_D(\eta) = \max_{1 \leq i \leq l} \nu_i(\eta)$. It is easy to see that ν_D is also a negligible function. Moreover, it immediately follows, from the definition of ν_D and A_i, that for all $i = 1, \ldots, l$, the confusion probability for distribution ensemble D_i is at most

$$\Pr\left[k \overset{R}{\leftarrow} \mathcal{K}(\eta), k_i \overset{R}{\leftarrow} \mathcal{K}_i(\eta), x \overset{R}{\leftarrow} D_i(\eta) : \mathcal{D}_k(\mathcal{E}_{k_i}(x)) \neq \bot)\right] \leq \nu_i(\eta) \leq \nu_D(\eta).$$

▮

5.2 The completeness theorem

Our main result is a converse to Theorem 3.1 for the case when the encryption scheme Π is a secure authenticated (or even just confusion free) encryption scheme. In conjunction with the result of [2], we prove the following:

[3]In fact, we are using an even weaker notion of authenticity than the one described in Definition 5.3. The notion described in the definition is called authenticity under *Chosen Plaintext Attack* (CPA), and is often regarded as the weakest useful notion for cryptographic applications, as opposed to authenticity under *Chosen Ciphertext Attack* (CCA) where the adversary is given access to both an encryption and decryption oracle. Here we are using authenticity under "No message attack", an even weaker notion than CPA, where the adversary is not given access to any oracle at all, and has to come up with a valid ciphertext with respect to a random unknown key.

Theorem 5.5 Let Π be a type-0 secure authenticated encryption scheme, or more generally a type-0 secure, confusion free encryption scheme. If E_0 and E_1 are acyclic expressions, then

$$E_0 \cong E_1 \text{ if and only if } [\![E_0]\!]_{\Pi(\eta)} \approx [\![E_1]\!]_{\Pi(\eta)}$$

The "only if" direction is the soundness result of [2]. Here we prove the "if" part.

Proof idea. We will show that from a bit string representation of E, i.e., a sample $e \overset{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$, one can recover the pattern of E, at least with high probability. To this end, we introduce functions that correspond to $\mathsf{F_{kr}}$ and $\mathsf{recoverable}$, but which operate on bit strings rather than on expressions from **Exp**. Then, we show that with overwhelming probability, the pattern recovered from the bit string is equivalent up to renaming to the pattern recovered from the corresponding expression. An algorithm for distinguishing between the distribution associated to two expressions E_0 and E_1 with different patterns, can be easily constructed: given a sample e from one of the two distributions $[\![E_0]\!]_{\Pi(\eta)}$ or $[\![E_1]\!]_{\Pi(\eta)}$, compute the pattern associated to e and check if it is equivalent to the pattern of E_0; if yes output 0, or output 1 otherwise. It is a simple consequence of the above discussion that such an algorithm will distinguish the two distributions with overwhelming probability. (See proof at the end of this section for details.)

Recovering the keys. Let E be a fixed expression, $e \in [\![E]\!]_{\Pi(\eta)}$ a random sample from the corresponding distribution, and τ the underlying key assignment. We claim that from bit string e one can recover the set

$$\tau(\mathsf{recoverable}(E)) = \{\tau(K) \mid K \in \mathsf{recoverable}(E)\}$$

i.e., the set of key values assigned to the keys in $\mathsf{recoverable}(E)$, with overwhelming probability.

In Figure 2, we define the functions $\mathsf{C_{kr}} : \mathsf{String} \times \mathcal{P}(\mathsf{Key}) \to \mathcal{P}(\mathsf{Key})$ and $\mathsf{Crecoverable} : \mathsf{String} \to \mathcal{P}(\mathsf{Key})$ with the intention of showing that they are the computational counterparts of $\mathsf{F_{kr}}$ and $\mathsf{recoverable}$ respectively. The algorithm $\mathsf{C_{kr}}$ takes as input a bit string e and a set of key values tT, and computes a set of key values by a simple case analysis on the type of e. The definition closely resembles the one of $\mathsf{F_{kr}}$. The interesting case is when e is a ciphertext. In this case, the algorithm attempts to decrypt e with each of the keys in tT. If the decryption does not succeed for any of the keys in tT, then it output tT. Otherwise, if $p \neq \perp$ is the result of the decryption for some key $k \in tT$, the algorithm outputs the result of the recursive call $\mathsf{C_{kr}}(p, tT)$.

The algorithm $\mathsf{Crecoverable}$, on bit string e as input computes the set of *all* key values that are recoverable from e. The definition of $\mathsf{Crecoverable}$ closely follows the definition of $\mathsf{recoverable}$. It iteratively computes an increasing sequence of key value sets, $T_0 \subseteq T_1 \subseteq ...$ where $T_{i+1} = \mathsf{C_{kr}}(e, T_i)$. The iteration stops when $T_{i+1} = T_i$. We now proceed to show that functions $\mathsf{F_{kr}}, \mathsf{recoverable}$ and $\mathsf{C_{kr}}, \mathsf{Crecoverable}$ are indeed related. More

```
Algorithm $\mathsf{C_{kr}}(e, tT)$
    if $e = \langle b, \text{"}block\text{"}\rangle$ output $tT$
    if $e = \langle k, \text{"}key\text{"}\rangle$ output $tT \cup \{k\}$
    if $e = \langle (m, n), \text{"}pair\text{"}\rangle$ output $\mathsf{C_{kr}}(m, tT) \cup \mathsf{C_{kr}}(n, tT)$
    if $e = \langle c, \text{"}ciphertext\text{"}\rangle$ then
        if for exactly one $k \in tT, \mathcal{D}_k(y) \neq \perp$
            then output $\mathsf{C_{kr}}(\mathcal{D}_k(c), tT)$
            else output $tT$
```

```
Algorithm $\mathsf{Crecoverable}(e)$
    $T_0 \leftarrow \emptyset$; $cont \leftarrow true$
    while $cont$
        $T_{i+1} \leftarrow \mathsf{C_{kr}}(e, T_i)$
        if $T_{i+1} = T_i$ then $cont \leftarrow false$
            else $i \leftarrow i + 1$
    output $T_i$
```

Figure 2: The algorithms $\mathsf{C_{kr}}$ and $\mathsf{Crecoverable}$ define computational counterparts to the functions $\mathsf{F_{kr}}$ and $\mathsf{recoverable}$. The input e of the algorithms is a sample from $[\![E]\!]_{\Pi(\eta)}$ for some E

precisely, if $T \subseteq$ **Keys** is a set of key symbols and $e \overset{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$, we will show that $\tau(\mathsf{F_{kr}}(E, T)) = \mathsf{C_{kr}}(e, \tau(T))$ and $\tau(\mathsf{recoverable}(E)) = \mathsf{Crecoverable}(e)$, except with negligible probability. This is formally captured by the following two lemmas.

Lemma 5.6 Let Π be a confusion free encryption scheme and let $E \in$ **Exp**, $T \subseteq$ **Keys**. If e is a bit string selected according to the distribution $[\![E]\!]_{\Pi(\eta)}$ and τ is the underlying key assignment, then

$$\Pr\left[\mathsf{C_{kr}}(e, \tau(T)) \neq \{\tau(\mathsf{F_{kr}}(E, T)\})\right] \leq |E||T|\nu(\eta) \tag{4}$$

for some negligible function ν.

Proof: Let ν be the negligible function associated to expression E, as defined in Remark 5.2. We prove the statement of the lemma by induction on the structure of E. For each case, we consider the sets $\mathsf{F}_{\mathsf{kr}}(E,T)$ and $\mathsf{C}_{\mathsf{kr}}(e,\tau(T))$ and then compute the probability from the statement of the lemma. In the following case analysis on the type of e, the probabilities are taken over the random choice of e, or, equivalently, the choice of the underlying key assignment τ.

- If $E = B$ for some $B \in \mathbf{Block}$ then e is of the form $\langle b, \text{“}\mathit{block}\text{”}\rangle$, where b is the intended representation for block B. It follows from the definitions of F_{kr} and C_{kr} that in this case $\mathsf{F}_{\mathsf{kr}}(E,T) = T$ and $\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) = \tau(T) = \tau(\mathsf{F}_{\mathsf{kr}}(E,T))$. Therefore:

$$\Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) \neq \{\tau(\mathsf{F}_{\mathsf{kr}}(E,T)\})\right] = 1 - \Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) = \tau(\mathsf{F}_{\mathsf{kr}}(E,T))\right] = 0$$

- If $E = K$ then e is of the form $\langle \tau(K), \text{“}\mathit{key}\text{”}\rangle$. From the definitions of F_{kr} and C_{kr} it follows that

$$\mathsf{F}_{\mathsf{kr}}(E) = \{K\} \cup T \text{ and } \mathsf{C}_{\mathsf{kr}}(e,\tau(T)) = \tau(T) \cup \tau\{K\} = \tau(T \cup \{K\}) = \tau(\mathsf{F}_{\mathsf{kr}}(E,T))$$

  Therefore:

$$\Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E,T))\right] = 1 - \Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) = \tau(\mathsf{F}_{\mathsf{kr}}(E,T))\right] = 0$$

- If $E = (E_0, E_1)$ then e is of the form $e = \langle (e_0, e_1), \text{“}\mathit{pair}\text{”}\rangle$, where $e_i \xleftarrow{R} [\![E_i]\!]_{\Pi(\eta)}, i = 0, 1$. (Notice that although the conditional distributions of e_0 and e_1 equal the distribution associated to the corresponding subexpressions, the two distributions are not independent because the are obtained by the same random key assignment. However, this does not affect the validity of the proof below.) From the definition of F_{kr} and C_{kr}, it follows that

$$\mathsf{F}_{\mathsf{kr}}(E) = \mathsf{F}_{\mathsf{kr}}(E_0,T) \cup \mathsf{F}_{\mathsf{kr}}(E_1,T) \text{ and } \mathsf{C}_{\mathsf{kr}}(E,T) = \mathsf{C}_{\mathsf{kr}}(E_1,T) \cup \mathsf{C}_{\mathsf{kr}}(E_2,T)$$

  Therefore (the probabilities are over the choices of the common key assignment τ) we have the following inequalities:

$$\begin{aligned}
&\Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E,T))\right] \\
&\quad\leq \Pr\left[\mathsf{C}_{\mathsf{kr}}(e_0,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E_0,T)) \text{ or } \mathsf{C}_{\mathsf{kr}}(e_1,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E_1,T))\right] \\
&\quad\leq \Pr\left[\mathsf{C}_{\mathsf{kr}}(e_0,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E_0,T))\right] + \Pr\left[\mathsf{C}_{\mathsf{kr}}(e_1,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E_1,T))\right])
\end{aligned}$$

  Using the induction hypothesis:

$$\Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) \neq \tau(\mathsf{F}_{\mathsf{kr}}(E,T))\right] \leq |E_0||T|\nu(\eta) + |E_1||T|\nu(\eta) = (|E_0| + |E_1|)|T|\nu(\eta) = |E||T|\nu(\eta)$$

- If $E = \{E_0\}_K$ then e is of the form $\langle \mathcal{E}_{\tau(K)}(e_0), \text{“}\mathit{ciphertext}\text{”}\rangle$ for some $e_0 \xleftarrow{R} [\![E_0]\!]_{\Pi(\eta)}$. We distinguish between the cases when the key K is in T or not.

  1. If $K \not\in T$ then according to the definition of F_{kr} we have that $\mathsf{F}_{\mathsf{kr}}(E,T) = T$. From the definition of C_{kr}, in order for the set $\mathsf{C}_{\mathsf{kr}}(e,\tau(T))$ to be different from $\tau(T)$, decrypting e_0 with exactly one key in $\tau(T)$ must be successful. This however is prevented by the security properties of Π. Formally:

$$\begin{aligned}
\Pr\left[\mathsf{C}_{\mathsf{kr}}(e,\tau(T)) \neq \{\tau(\mathsf{F}_{\mathsf{kr}}(E,T)\})\right] &\leq \Pr\left[\exists k \in \tau(T).\mathcal{D}_k(\mathcal{E}_{\tau(K)}(e_0)) \neq \bot\right] \\
&\leq \sum_{k \in \tau(T)} \Pr\left[\mathcal{D}_k(\mathcal{E}_{\tau(K)}(e_0))) \neq\bot\right]
\end{aligned}$$

Since $K \notin T$, $\tau(K)$ and and $k \in \tau(T)$ are independently generated, so from the confusion freeness of Π we have that $\Pr\left[\mathcal{D}_k(\mathcal{E}_{\tau(k)}(e_0) \neq \bot\right] \leq \nu(\eta)$. Thus, we obtain that

$$\Pr\left[\mathsf{C_{kr}}(e, \tau(T)) \neq \{\tau(\mathsf{F_{kr}}(E,T)\}\right] \leq |T|\nu(\eta) \leq \ |E||T|\nu(\eta)$$

2. If $K \in T$ then according to the definition of $\mathsf{F_{kr}}$ we have $\mathsf{F_{kr}}(\{E\}_K, T) = \mathsf{F_{kr}}(E,T)$. The following inequalities follow from the definition of $\mathsf{C_{kr}}$ (the probabilities are taken over the choice of e, or equivalently, the choice of e_0)

$$\begin{aligned}
&\Pr\left[\mathsf{C_{kr}}(e, \tau(T)) = \tau(\mathsf{F_{kr}}(E,T))\right] \geq \\
&\quad\geq \ \Pr\left[\mathsf{C_{kr}}(e_0, \tau(T)) = \tau(\mathsf{F_{kr}}(E_0,T)) \text{ and } \forall k_i \in \tau(T)\setminus\{\tau(K)\}.\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) = \bot\right] \\
&\quad= \ 1 - \Pr\left[\mathsf{C_{kr}}(e_0, \tau(T)) \neq \tau(\mathsf{F_{kr}}(E_0,T)) \text{ or } \exists k_i \in \tau(T)\setminus\{\tau(K)\}.\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \bot\right] \\
&\quad\geq \ 1 - \left(\Pr\left[\mathsf{C_{kr}}(e_0, \tau(T)) \neq \tau(\mathsf{F_{kr}}(E_0,T))\right] + \sum_{k_i \in \tau(T)\setminus\{\tau(K)\}} \Pr\left[\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \bot\right]\right)
\end{aligned}$$

Since Π is a secure authenticated scheme it follows that for any $k_i \in \tau(T)\setminus\{\tau(K)\}$ the inequality $\Pr\left[\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \bot\right] \leq \nu(\eta)$ holds. Using the induction hypothesis

$$1 - (|E_0||T|\nu(\eta) + (|T|-1)\nu(\eta)) \geq 1 - (|E_0|+1)|T|\nu(\eta) \geq 1 - |E||T|\nu(\eta)$$

we obtain that

$$\Pr\left[\mathsf{C_{kr}}(e, \tau(T)) \neq \{\tau(\mathsf{F_{kr}}(E,T)\}\right] \leq \ |E||T|\nu(\eta)$$

▮

Lemma 5.7 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a confusion free encryption scheme and let $E \in \mathbf{Exp}$. If e is a bit string selected according to the distribution $[\![E]\!]_{\Pi(\eta)}$ and τ is the underlying key assignment, then

$$\Pr\left[\mathsf{Crecoverable}(e) \neq \tau(\mathsf{recoverable}(E))\right] \leq |E|^3\nu(\eta)$$

for some negligible function $\nu(\cdot)$.

Proof: Let ν be the negligible function associated to E as defined in Remark 5.2. We prove by induction that the statement of the lemma holds. Consider the sets $\emptyset = G_0 \subseteq G_1 \subseteq \ldots$ used to define $\mathsf{recoverable}$ (i.e. $G_{i+1} = \mathsf{F_{kr}}(E, G_i)$) and the sets $\emptyset = T_0 \subseteq T_1 \subseteq \ldots$ computed during the execution of $\mathsf{Crecoverable}$ (i.e. $T_{i+1} = \mathsf{C_{kr}}(e, T_i)$). We denote by A_i the event that at iteration i, $T_i = \tau(G_i)$. Note that A_0 is vacuously true. Then we have the following:

$$\Pr[A_{i+1}] \geq \Pr[A_{i+1} \wedge A_i] = \Pr[\, A_{i+1} \mid A_i\,]\Pr[A_i] = (1 - \Pr[\,\neg A_{i+1} \mid A_i\,])\Pr[A_i]$$

Replacing all A_i's and using Lemma 5.6 we obtain:

$$\begin{aligned}
&\Pr[T_{i+1} = \tau(G_{i+1})] \\
&\quad= \ (1 - \Pr[\,\mathsf{C_{kr}}(e, T_i) \neq \mathsf{F_{kr}}(E, G_i)) \mid T_i = \tau(G_i)\,])\Pr[T_i = \tau(G_i)] \\
&\quad\geq \ (1 - |E||G_i|\nu(\eta))\Pr[T_i = \tau(G_i)] \\
&\quad\geq \ (1 - |E|^2\nu(\eta))\Pr[T_i = \tau(G_i)]
\end{aligned}$$

The above inequality was proved for arbitrary i, so by repeating the above for $i-1, i-2, ...$ etc. we obtain:

$$\Pr[T_{i+1} = \tau(G_{i+1})] \geq (1 - |E|^2\nu(\eta))^i\Pr[T_0 = \tau(G_0)] = (1 - |E|^2\nu(\eta))^i \geq 1 - i|E|^2\nu(\eta)$$

Since the total number of keys in E is obviously bounded by $|E|$, and at each iteration the algorithm Crecoverable recovers at least one key or it stops, $i \le |E|$ and therefore. $\Pr[T_i \ne \tau(G_i(E))] \le |E|^3\nu(\eta)$ which immediately implies the conclusion of the lemma:

$$\Pr[\mathsf{Crecoverable}(E) \ne \tau(\mathsf{recoverable}(E))] \le |E|^3\nu(\eta)$$

▌

Pattern computation. We have shown that given $e \overset{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$ we can compute the key values assigned to the keys in recoverable(E) with non-negligible probability. Our next step is to show that we can associate a computational counterpart to the function p defined in Section 3.

Consider the function $\mathsf{psp} : \mathsf{String} \times \mathcal{P}(\mathsf{Key}) \to \mathbf{Pattern}$ shown in Figure 3. On input a string e, and a set of key values tT it returns a pattern. The algorithm uses an arbitrary, but fixed, *naming function* f, which associates to each key value a unique key name. E.g., f might sort all values in tT in lexicographic order, and assign symbols $K_1, K_2, \ldots, K_{|T|}$ to the key values according to their position in the sorted list. We also assume that it is easy to recover a block symbol B from its bit string representation. The algorithm makes a simple case analysis on the type of e. Again, the only nontrivial case is when e is a ciphertext, in which case the algorithm attempts to decrypt e using each of the keys in tT. If the decryption does not succeed, it outputs $\Box$. If the decryption succeeds for some key k, then due to the confusion freeness of Π the key k was most probably used to compute the ciphertext, in which case the algorithm proceeds recursively.

```
Let $f : tT \to \mathbf{Keys}$ be an injective function

Algorithm $\mathsf{psp}(e, tT)$
    if $e = \langle b, \text{“}block\text{”}\rangle$ output block symbol $B$ corresponding to $b$;
    if $e = \langle k, \text{“}key\text{”}\rangle$ and $k \in tT$ then output $f(k)$;
    if $e = \langle (m,n), \text{“}pair\text{”}\rangle$ output $(\mathsf{psp}(m, tT), \mathsf{psp}(n, tT))$;
    if $e = \langle c, \text{“}ciphertext\text{”}\rangle$
        if for exactly one key $k \in tT, \mathcal{D}_k(c) \ne \perp$
        output $\{\mathsf{psp}(\mathcal{D}_k(c), tT)\}_{f(k)}$
        else output $\Box$;
```

Figure 3: The computational counterpart of pattern function p

The next lemma states that the “syntactic” function p and its “computational” counterpart psp have essentially the same behavior.

Lemma 5.8 Let Π be a confusion free encryption scheme and let $E \in \mathbf{Exp}$. Let $T \subseteq \mathbf{Keys}$ be a set of key symbols. If e is a bit string selected according to the distribution $[\![E]\!]_{\Pi(\eta)}$, and τ is the underlying key assignment then

$$\Pr[\mathsf{psp}(e, \tau(T)) \not\cong \mathsf{p}(E, T)] \le |E||T|\nu(\eta)$$

for some negligible function $\nu(\cdot)$.

Proof: We will prove the statement of the lemma with ν equal to the negligible function associated to E defined in Remark 5.2. Let $\tau : \mathbf{Keys} \to \mathsf{String}$ be the key assignment underlying $e \in [\![E]\!]_{\Pi(\eta)}$. Consider the injective key renaming function $\sigma\colon K \mapsto f(\tau(K))$, where f is the naming function used in psp. We will show that the following result holds:

$$\Pr[\mathsf{psp}(e, \tau(T)) \ne \mathsf{p}(E, T)\sigma] \le |E||T|\nu(\eta)$$

where $(E,T)\sigma$ is the pair $(E\sigma, T)$ obtained by applying the renaming σ to expression E. The proof is by induction on the structure of E.

- If $E = B$ then $e = \langle b, \text{"}\mathit{block}\text{"}\rangle$ and $\mathsf{p}(E,T) = b$, where b is the bit string representation of B, we obtain:

$$\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] = 1 - \Pr[\mathsf{psp}(e,\tau(T)) = \mathsf{p}(E,T)\sigma] = 1 - \Pr[b = b\sigma] = 0;$$

- If $E = K$ for some $K \in \mathbf{Keys}$ then $e = \langle \tau(K), \text{"}\mathit{key}\text{"}\rangle$ and $\mathsf{p}(E,T) = K$ and we obtain:

$$\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] = 1 - \Pr[f(\tau(K)) = K\sigma] = 1 - \Pr[f(\tau(K)) = f(\tau(K))] = 0$$

- If $E = (E_0, E_1)$ for some $E_0, E_1 \in \mathbf{Exp}$; then $e = \langle (e_0, e_1), \text{"}\mathit{pair}\text{"}\rangle$ with $e_i \stackrel{R}{\leftarrow} [\![E_i]\!]_{\Pi(\eta)}$ for $i = 0, 1$ and $\mathsf{p}(E,T) = (\mathsf{p}(E_0,T), \mathsf{p}(E_1,T))$. The following equalities (the probabilities are taken on the choice of $e \stackrel{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$, or alternatively on the choices of $e_0 \stackrel{R}{\leftarrow} [\![E_0]\!]_{\Pi(\eta)}$ and $e_1 \stackrel{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}$), hold with respect to a common key assignment τ.

$$\begin{aligned}\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] &\leq \Pr[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,T)\sigma \text{ or } \mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,T))\sigma] \\ &\leq \Pr[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,T)\sigma] + \Pr[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,T))\sigma].\end{aligned}$$

  Using the induction hypothesis we obtain:

$$\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] \leq |E_0||T|\nu(\eta) + |E_1||T|\nu(\eta) = (|E_0| + |E_1|)|T|\nu(\eta) = |E||T|\nu(\eta)$$

- If $E = \{E_0\}_K$ then $e = \langle \mathcal{E}_{\tau(K)}(e_0), \text{"}\mathit{ciphertext}\text{"}\rangle$, with $e_0 \stackrel{R}{\leftarrow} [\![E_0]\!]_{\Pi(\eta)}$.
  We distinguish between the cases when the key K is in the set T of known keys or not:

  1. If $K \notin T$, then $\mathsf{p}(E,T) = \Box$ and we obtain

$$\Pr[\mathsf{psp}(e,\tau(T)) = \mathsf{p}(E,T)\sigma] = \Pr\left[(\forall k_i \in \tau(T))\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) = \perp\right]$$

     which in turn implies

$$\begin{aligned}&\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] \\ &\quad= \Pr\left[(\exists k_i \in \tau(T))\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \perp\right] \\ &\quad\leq \sum_{k_i \in \tau(T)} \Pr\left[\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0))\right] \leq |T|\nu(\eta) \leq |E||T|\nu(\eta)\end{aligned}$$

  2. If $K \in T$ then $\mathsf{p}(E,T) = \{\mathsf{p}(E_0,K)\}_K$, and the following equalities hold:

$$\begin{aligned}&\Pr[\mathsf{psp}(e,\tau(T)) = \mathsf{p}(E,T)\sigma] \\ &\quad\geq \Pr\left[\mathsf{psp}(e_0,\tau(T)) = \mathsf{p}(E_0,K) \text{ and } (\forall k_i \in \tau(T) \setminus \{\tau(K)\})\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) = \perp\right] \\ &\quad= 1 - \Pr\left[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,K) \text{ or } (\exists k_i \in \tau(T) \setminus \{\tau(K)\})\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \perp\right] \\ &\quad\geq 1 - (\Pr[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,K)] + \Pr\left[(\exists k_i \in \tau(T) \setminus \{\tau(K)\})\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \perp\right]) \\ &\quad\geq 1 - (\Pr[\mathsf{psp}(e_0,\tau(T)) \neq \mathsf{p}(E_0,K)] + \sum_{k_i \in \tau(T) \setminus \{\tau(K)\}} \Pr\left[\mathcal{D}_{k_i}(\mathcal{E}_{\tau(K)}(e_0)) \neq \perp\right]) \\ &\quad\geq 1 - (|E_0||T|\nu(\eta) + |T|\nu(\eta))\end{aligned}$$

     The last inequality is obtained by using the induction hypothesis together with the confusion freeness property of Π. It immediately follows that:

$$\Pr[\mathsf{psp}(e,\tau(T)) \neq \mathsf{p}(E,T)\sigma] \leq |E_0||T|\nu(\eta) + |T|\nu(\eta) \leq |E||T|\nu(\eta)$$

▮

The probability that $\mathsf{psp}(e, \mathsf{Crecoverable}(e)) \not\cong \mathsf{p}(E, \mathsf{recoverable}(E))$ can be computed using the above lemmas. Since this quantity is negligible, it follows that given $e \stackrel{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$ one can recover, with non-negligible probability the pattern of E as $\mathsf{psp}(e, \mathsf{Crecoverable}(e))$. We will use this to prove our main theorem.

Corollary 5.9 Let Π be a confusion free encryption scheme, and let $E, E' \in \mathbf{Exp}$ be two expressions such that $\mathsf{pattern}(E) \not\cong \mathsf{pattern}(E')$. If e is a bit string sampled from the distribution $[\![E]\!]_{\Pi(\eta)}$ then:

$$\Pr[tT \leftarrow \mathsf{Crecoverable}(e) : \mathsf{pattern}(E) \not\cong \mathsf{psp}(e, tT)] \leq 2|E|^3\nu(\eta) \tag{5}$$

and

$$\Pr[tT \leftarrow \mathsf{Crecoverable}(e) : \mathsf{pattern}(E') \not\cong \mathsf{psp}(e, tT)] \geq 1 - 2|E|^3\nu(\eta) \tag{6}$$

for some negligible function ν.

Proof: First, note that equation (6) is an immediate consequence of equation (5). Indeed, using that $\mathsf{pattern}(E) \not\cong \mathsf{pattern}(E')$ we have:

$$\Pr[tT \leftarrow \mathsf{Crecoverable}(e) : \mathsf{pattern}(E') \not\cong \mathsf{psp}(e, tT)] \geq \Pr[\mathsf{pattern}(E) \cong \mathsf{psp}(e, tT)] \geq 1 - 2|E|^3\nu(\eta)$$

It remains to show that equation (5) is true. This can be seen as follows (let $\nu(\cdot)$ be the function associated to E as in Remark 5.2):

$$\begin{aligned}
&\Pr[tT \leftarrow \mathsf{Crecoverable}(e) : \mathsf{pattern}(E) \cong \mathsf{psp}(e, tT)] \\
&\quad \geq \quad \Pr[tT = \tau(\mathsf{recoverable}(E)) \wedge \mathsf{pattern}(E) \cong \mathsf{psp}(e, tT)] \\
&\quad = \quad \Pr[\ \mathsf{p}(E, \mathsf{recoverable}(E)) \cong \mathsf{psp}(e, tT) \mid tT = \tau(\mathsf{recoverable}(E))\ ] \cdot \Pr[tT = \tau(\mathsf{recoverable}(E))] \\
&\quad = \quad \Pr[\mathsf{p}(E, \mathsf{recoverable}(E)) \cong \mathsf{psp}(e, \tau(\mathsf{recoverable}(E)))] \cdot \Pr[\mathsf{Crecoverable}(e) = \tau(\mathsf{recoverable}(E))]
\end{aligned}$$

We bound each factor of the product by using Lemma 5.6 and Lemma 5.7 respectively, and obtain:

$$\Pr[tT \leftarrow \mathsf{Crecoverable}(e) : \mathsf{pattern}(E) \cong \mathsf{psp}(e, tT)] \geq (1 - |E|^2\nu(\eta)) \cdot (1 - |E|^3\nu(\eta)) \geq 1 - 2|E|^3\nu(\eta)$$

Equation (5) follows. ▮

Proof of Theorem 5.5 For any two expressions E_0 and E_1 such that $\mathsf{pattern}(E_0) \not\cong \mathsf{pattern}(E_1)$ we show that there exists an algorithm that has non-negligible advantage in distinguishing between $[\![E_0]\!]_{\Pi(\eta)}$ and $[\![E_1]\!]_{\Pi(\eta)}$. We propose the following distinguisher $D(e, \eta)$ where e is a sample from one of the distributions $[\![E_0]\!]_{\Pi(\eta)}$ or $[\![E_1]\!]_{\Pi(\eta)}$ and η is the security parameter. Compute the pattern $\mathsf{psp}(e, \mathsf{Crecoverable}(e))$ and compare it with the pattern of E_0. This can be done in polynomial time by a simple unification-like algorithm. If they are equivalent (up to renaming), then the algorithm outputs 0, otherwise it outputs 1. The distinguishing power of this algorithm follows by applying equations (5) and (6) from Corollary 5.9. More precisely, the advantage of algorithm A in distinguishing between $[\![E_0]\!]_{\Pi(\eta)}$ and $[\![E_1]\!]_{\Pi(\eta)}$ is:

$$\Pr\left[e \stackrel{R}{\leftarrow} [\![E_0]\!]_{\Pi(\eta)} : D(e, \eta) = 0\right] - \Pr\left[e \stackrel{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)} : D(e, \eta) = 0\right] \geq$$

$$(1 - 2|E_0|^3\nu(\eta)) - 2|E_1|^3\nu(\eta) = 1 - 2(|E_0|^3 + |E_1|^3)\nu(\eta)$$

where $2(|E_0|^3 + |E_1|^3)\nu(\eta)$ is negligible for every expressions E_0, E_1 of size polynomially bounded by the security parameter η.

6 An extension to practical encryption schemes

The pattern semantics of [2] does not take into account the size of encrypted messages. For example, the expressions $\{((((0,0),(1,1)),(0,0),(1,1)),0)\}_K$ and $\{0\}_K$ have the same pattern $\Box$. Thus, in order to achieve soundness, the encryption scheme Π has to satisfy the impractical requirement that ciphertexts hide the length of their corresponding plaintexts.

In this section we explore a refinement of the Abadi-Rogaway logic, already suggested in [2], that deals with the above length issue. In particular, the inequality of the two expressions defined above can be proved in the refined logic. The main idea is to replace the pattern for "indecipherable message", $\Box$, with a family of patterns, $\Box_n$, one for each $n \in \mathsf{N}$ in order to capture the notion of "undecryptable message of size n". To obtain our result, we consider a uniform behavior of the encryption scheme with respect to the length of the plaintext. We emphasize that the choice that we make is one out of the many possibilities, and it corresponds to popular encryption schemes as we will shortly explain.

We assume that the encryption scheme views the plaintext as a sequence of *basic message blocks*, and that a ciphertext is one message block longer than the corresponding plaintext. Practical symmetric encryption schemes such as CBC and CTR (*Cipher Block Chaining*, and *Counter* mode, see for example [4]) satisfy this property. We also assume that block symbols as well as key symbols are mapped to bit strings of size equal to one basic message block.

Consider the following *length function* $L : \mathbf{Exp} \to \mathsf{N}$ defined recursively by the following definitions:

1. $L(K) = 1$ for any $K \in \mathbf{Keys}$;
2. $L(B) = 1$ for any $B \in \mathbf{Blocks}$;
3. $L((E_0, E_1)) = L(E_0) + L(E_1)$ for any $E_0, E_1 \in \mathbf{Exp}$;
4. $L(\{E\}_K) = L(E) + 1$;

When applied to a formal expression E it returns the size of the expression. The definition is motivated by the above discussion: equation (4) says that the size of a ciphertext is one block bigger than the size of the underlying plaintext. The other equation have a straightforward interpretation. Patterns that capture the information regarding the length of plaintexts underlying undecryptable ciphertexts can now be easily defined. For this, we modify the pattern semantics function p given in Section 3.2 by replacing the equation $\mathsf{p}(\{E\}_K, T) = \Box$ with $\mathsf{p}(\{E\}_K, T) = \Box_{L(E)}$, for the case when $K \notin T$. For completeness, the entire definition is repeated below.

$$\begin{array}{ll}
\mathsf{p}(K,T) = K & \\
\mathsf{p}(B,T) = B & \\
\mathsf{p}((M,N),T) = (\mathsf{p}(M,T), \mathsf{p}(N,T)) & \\
\mathsf{p}(\{M\}_K,T) = \{\mathsf{p}(M,T)\}_K & (\text{if } K \in T) \\
\mathsf{p}(\{M\}_K,T) = \Box_{L(M)} & (\text{if } K \notin T)
\end{array}$$

Having patterns that capture length-related information allows a relaxation of the hypothesis of Theorem 5.5. Namely, we have the following:

Theorem 6.1 Let Π be a type-1 secure authenticated encryption scheme, or more generally a type-1 secure, confusion free encryption scheme. If If E_0 and E_1 are acyclic expressions, then

$$E_0 \cong E_1 \text{ if and only if } [\![E_0]\!]_{\Pi(\eta)} \approx [\![E_1]\!]_{\Pi(\eta)}$$

Proof:

The proof of the above theorem closely follows the one for Theorem 5.5. Here we sketch the outline. For the completeness result we apply the same strategy, i.e. we show that, how to recover the pattern of E given a sample from $[\![E]\!]_{\Pi(\eta)}$. In fact, the whole proof is unchanged except the pattern recovery algorithm (Figure 3.) The change reflects the modified pattern semantics in which $\Box$ has been replaced by $\{\Box\}_{n\in\mathsf{N}}$, and is as follows. By the assumption that Π is of type-1, there exists a function L which on input a ciphertext c returns $L(c)$, the block length of the underlying plaintext. Then, in the last step of the case analysis, when the input is a ciphertext c that cannot be decrypted, the output of psp is $\Box_{L(c)}$. A sequence of lemmas as for Theorem 5.5 yield the proof of completeness.

For the soundness result, recall that in the hybrid argument proof of [2], one important step was to show how to attach a distribution to a pattern. Since we have modified the definition of pattern semantics in order to take into account the length of the plaintexts, this needs to be changed. The modification is rather straightforward: the distribution associated to $\Box_n$ is obtained by running the encryption algorithm on $\mathbf{0}^n$, where $\mathbf{0}$ is some fixed element of **Block**, and $\mathbf{0}^n$ denotes of n identical copies of block $\mathbf{0}$. This is actually the only modification that is needed in order to transform the proof of [2] into a proof for soundness direction of Theorem 6.1. ▌

7 Eavesdropping adversaries

In this section we sketch how our results can be adapted to the extension of the Abadi Rogaway logic introduced in [1]. The new framework allows description of more complex protocols involving a finite set of parties communicating synchronously, using elements of **Exp**, over a network of internal and external channels $\mathbf{Channels} = \mathbf{Channels}_i \cup \mathbf{Channels}_e$. External channels represent communication lines that can be observed by an adversary. Internal channels are perfectly secure communication lines that can be used, for example, to maintain some state information, or to represent interaction between different parts of a program running on a single computer. Each party can potentially read messages from any channel, but can only write on a dedicated channel. This is without loss of generality, since more complex parties can be represented as a collection of such programs sharing their input channels. If we denote by P_c the program that writes on channel c, a protocol $\mathcal{P}$ is identified with a collection of programs $\{P_c | c \in \mathbf{Channels}\}$. The precise syntax of the language used to describe the programs is not relevant, and for simplicity we omit it. The intended execution model is to iteratively run the programs in $\mathcal{P}$, over an infinite sequence of time frames. The content of channel c at time $t+1$ is obtained by running program P_c on the content of the channels at time t. It is assumed that at time 0 the channels contain no data. Depending on how programs are interpreted (manipulating formal expressions, or actual bitstrings) a formal and a computational semantics can be associated to any protocol $\mathcal{P}$. We next present simplified versions of both the formal and computational semantics, which are nevertheless sufficient for explaining our extension.

Formal semantics. First, consider an abstract execution in which the programs manipulate expressions from **Exp**. For each time frame t, program $\mathcal{P}$ defines a channel configuration $[\![\mathcal{P}]\!]_t$ describing the content of the channels at time t. Formally $[\![\mathcal{P}]\!]_t$ is a function from **Channels** to **Exp**. The initial configuration $[\![\mathcal{P}]\!]_0$ associates to every channel the empty expression. Configurations $[\![\mathcal{P}]\!]_t$ for $t > 0$ are defined inductively, running the programs in $\mathcal{P} = \{P_c | c \in \mathbf{Channels}\}$. In configuration $[\![\mathcal{P}]\!]_t$, the content of channel c is the output of program P_c, evaluated on the expression defined by $[\![\mathcal{P}]\!]_{t-1}$. This way, program $\mathcal{P}$ defines an infinite *execution trace* $[\![\mathcal{P}]\!] = ([\![\mathcal{P}]\!]_n)_{n \in \mathsf{N}}$. A simple way to look at $\mathcal{P}$ is as an infinite sequence of tuples of expressions indexed by **Channels**.[4]

A rather straightforward lifting of the definition of a pattern to traces, yields a measure for the information an adversary obtains by observing the communication on the external channels. Specifically, for every configuration $[\![\mathcal{P}]\!]_t$, we define the external configuration $[\![\mathcal{P}]\!]_t^e$ as the restriction of $[\![\mathcal{P}]\!]_t$ to the channels in $\mathbf{Channels}_e$. Then, for any execution prefix $[\![\mathcal{P}]\!]_{\leq n}^e = ([\![\mathcal{P}]\!]_0^e, \ldots, [\![\mathcal{P}]\!]_n^e)$, we compute a pattern $\mathsf{pattern}([\![\mathcal{P}]\!]_{\leq n}^e$ in the usual way: we first define the recoverable keys (at time n) $T_n = \mathsf{recoverable}([\![\mathcal{P}]\!]_{\leq n}^e)$ applying the key recovery function to the concatenation of all messages sent over the external channels during the first n steps of the computation, and then we use the keys T_n to compute the pattern

$$\mathsf{pattern}([\![\mathcal{P}]\!]_{\leq n}^e) = (\mathsf{p}([\![\mathcal{P}]\!]_1^e, T_n), \mathsf{p}([\![\mathcal{P}]\!]_2^e, T_n), \ldots, \mathsf{p}([\![\mathcal{P}]\!]_n^e, T_n)).$$

Two (infinite) traces $\mathbf{t}_1$ and $\mathbf{t}_2$ are said to be equivalent, denoted $\mathbf{t}_1 \cong \mathbf{t}_2$, if all their prefixes of equal length yield equal patterns (up to renaming). An informal interpretation of the definition is that two systems are said to be equivalent when an adversary cannot tell them apart by observing any finite partial executions.[5]

[4]For simplicity we did not include in our presentation the key and coin renaming mechanism that ensures that new keys are represented by different symbols for different invocations of the programs P_c.

[5]Our definition is an alternative to the approach of [1], where to each program a single infinite trace is associated, and the equivalence is defined in terms of these infinite sequences. Nevertheless the equivalence relation induced on protocols is the same.

Two protocols $\mathcal{P}$ and $\mathcal{Q}$ are said to be equivalent if their traces are equivalent. Formally, we say $\mathcal{P} \cong \mathcal{Q}$ iff $[\![\mathcal{P}]\!] \cong [\![Q]\!]$.

The following subtle point is important for the extension of our result. The syntax of the language allows describing protocols that forward ciphertexts from one channel to another. Since in real executions an adversary can check equality of ciphertexts through bit by bit comparison, this kind of information has to be captured by the formal semantics. In [1] this is done by replacing the symbol $\Box$ (of the Abadi Rogaway logic) with a family of symbols $\Box_r$, one for each symbol r (used to represent randomness used to create a ciphertext). Since the syntax of the language does not allow the use of the same randomness to create different ciphertexts, two occurrences of $\Box_r$ within the same trace, must represent identical ciphertexts.

COMPUTATIONAL SEMANTICS. We now consider a "real" execution of protocol $\mathcal{P}$, in which the channels contain bitstrings, keys are obtained by running the key generator algorithm, and formal encryption is instantiated with an actual encryption scheme Π. Let η be some fixed security parameter and assume that each of the channels contains some (possibly empty) bit string. Running *once* the programs in $\mathcal{P} = \{P_c | c \in \mathbf{Channels}\}$ defines a probability distribution (and thus a probability ensemble) on the content of the channels, where the distribution on channel c is equal to the output distribution of program P_c (when new randomness and new keys are generated according to the security parameter η). Notice that distributions associated to different channels are not necessarily independent because the programs may share input channels.

Starting in the all empty channels configuration, one can associate to each protocol $\mathcal{P}$ an infinite sequence of probability ensembles $[\![\mathcal{P}]\!]^e_1, [\![\mathcal{P}]\!]^e_2, \ldots$. We let $[\![\mathcal{P}]\!]_t$ denote the probability ensemble describing the content of the channels after iterating the programs in $\mathcal{P}$ for t times. Starting from the distribution $[\![\mathcal{P}]\!]_0$ that associate the empty string to every channel with probability one, for each t distribution $[\![\mathcal{P}]\!]_t$ is obtained from $[\![\mathcal{P}]\!]_{t-1}$ as described above. We let $[\![\mathcal{P}]\!]^e_t$ denote the probability ensemble restricted to the external channels[6].

Computational equivalence of two protocols $\mathcal{P}$ and $\mathcal{Q}$, is defined by requiring that finite executions of the protocols cannot be told apart by observing the external channel:

$$\mathcal{P} \approx_\Pi \mathcal{Q} \text{ if and only if } (\forall i \in \mathsf{N})[\![\mathcal{P}]\!]^e_{\leq i} \approx [\![\mathcal{Q}]\!]^e_{\leq i},$$

where $[\![\mathcal{P}]\!]^e_{\leq n}$ denotes $([\![\mathcal{P}]\!]_1, \ldots, [\![\mathcal{P}]\!]_n)$.

COMPLETENESS THEOREM. The following soundness theorem relates the two semantics:

Theorem 7.1 [Abadi-Jürjens] Let $\mathcal{P}$ and $\mathcal{Q}$ be two systems that do not generate encryption cycles. Suppose Π is a type-0 secure and confusion-free encryption scheme. If $\mathcal{P} \cong \mathcal{Q}$ then $\mathcal{P} \approx_\Pi \mathcal{Q}$.

Note that the hypothesis already contains the requirement that Π is a confusion free encryption scheme. This is necessary because the formalism explicitly uses decryption, and therefore some mechanism for checking when the decryption was done with the right key is also needed.

The authors of [1] also mention that the converse is likely to be true, but no proof is given. We will explain how our techniques can be extended to proof that completeness also holds. The basic intuition is the same as for the Abadi Rogaway setting: given a sample from distribution $[\![\mathcal{P}]\!]^e_{\leq i}$ associated to executing $\mathcal{P}$, one can recover the pattern corresponding of the ith prefix of the execution trace. This can be done by adopting the strategy of Section 5 to samples coming from distributions associated to channels (as opposed to expressions.) To show the logic is complete, assume that $[\![\mathcal{P}]\!] \not\approx_\Pi [\![\mathcal{Q}]\!]$, i.e. for some i, $[\![\mathcal{P}]\!]^e_{\leq i} \not\approx [\![\mathcal{Q}]\!]^e_{\leq i}$. If this is the case, then the ith prefixes of $[\![\mathcal{P}]\!]$ and $[\![\mathcal{Q}]\!]$, have different patterns, so $[\![P]\!] \not\cong [\![\mathcal{Q}]\!]$.

While the details may be technically involved, the proof is conceptually very simple. To implement key recovery, the algorithm given in Figure 2 needs little change: it only needs to be adapted to work for samples coming from distributions associated to executions rather than simple expressions. The changes to the pattern function (Figure 3) are slightly more involved. Besides the above issue, one also needs to detect undecryptable ciphertext duplicates (see the remark regarding formal semantics for traces). This can

[6]In [1], these probabilities ensembles are carefully defined in order to take care of the case when certain keys and random strings need to have the same value during different time periods. Technically, this is realized using *choice functions* that assign values to the keys and to the random coins used during each time period

be realized by using a list $L = \{(c_1, r_1), ..., (c_k, r_k)\}$ of ciphertext-randomness pairs: when the input to the pattern function is a ciphertext $\bar{c}$, the algorithm first examines the list to check if $c_i = \bar{c}$ for some i. If this is the case then the algorithm outputs $\Box_{r_i}$ (the ciphertext has already been seen). Otherwise, it adds the pair $(\bar{c}, r_{k+1})$ to the list and outputs $\Box_{r_{k+1}}$. The remaining details are straightforward, and left to the reader.

8 Conclusion

In this paper we analyzed the completeness of the Abadi-Rogaway logic of encrypted expressions, and considered various extensions of the basic logic. In particular, we considered logics that allow to model realistic encryption functions that do not hide the length of the message being transmitted, and complex protocols of distributed programs communicating over a synchronous network as studied in [1]. All these logics are both sound and complete when the encryption scheme used to implement the protocols satisfies the appropriate notion of security. (Namely, indistinguishability and confusion-freeness.) In other words, the patterns associated to two programs by these logics are equivalent if and only if no efficient (probabilistic polynomial time) adversary can distinguish the messages transmitted by one or the other protocol with non-negligible advantage.

The cryptographic primitive considered in this paper is symmetric encryption. However, as already remarked in [2], all results can be easily extended to asymmetric (public key) encryption schemes as well. It should be possible also to extend the results of [2, 1] and this paper to other cryptographic primitives, whose definition is based on the notion of indistinguishability. Natural candidates are pseudorandom generators and pseudorandom functions. Extending these results to these primitives, as well as simple operations on messages (like exclusive-or of strings, string concatenation, or equality test) would be interesting because using these operations and cryptographic primitives it is already possible to model some interesting cryptographic protocols. (For example, the encryption scheme described in Section 4 can be expressed in terms of these operations.)

A natural question is whether the simple logic approach studied in this paper can be extended to other cryptographic primitives, like collision resistant hashing, message authentication codes, or digital signatures, whose security cannot be defined in terms of indistinguishability of distributions. This highlights what we believe is the main limitation of the approach of [2, 1]: these logics only capture *passive* attacks, where an adversary intercepts the messages exchanged by the parties participating in the protocol, but is unable to alter the messages or inject new messages in the network. Moreover, all parties are trusted to follow the protocol. Cryptographic primitives like digital signatures and hash functions (or even more complicated primitives like zero knowledge proofs) are mostly intended to avoid message tampering or deviations from the protocol. So the study of these primitives is interesting in a context where messages sent over a network can be altered by an adversary, or some parties may be malicious. Designing formal methods that can be used to reason about these more general adversarial behaviors, and still can be proved sound (and possibly complete) with respect to standard computational security notions, is a very interesting open problem.

8.1 Acknowledgments

We would like to thank Alexandra Boldyreva (a.k.a. Sasha) and Chanathip Namprempre (a.k.a. Meaw) for useful discussions.

References

[1] M. Abadi and J. Jürgens. Formal eavesdropping and its computational interpretation. In N. Kobayashi and B. Pierce, editors, *Proceedings of the Fourth International Symposium on Theoretical Aspects of Computer Software*, volume 2215 of *LNCS*, pages 82–94, Sendai, Japan, 29–31 Oct. 2001. Springer-Verlag.

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

[3] J. An and M. Bellare. Does encryption with redundancy provide authenticity? In *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 512–528, Berlin, Germany, 2001. Springer-Verlag.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, 1998.

[5] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology — ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 2000.

[6] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Fast Software Encryption*, LNCS. Springer-Verlag, 2001.

[7] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[9] J. Guttman, F. J. Thayer, and L. Zuck. The faithfulness of abstract encryption. In *Proceedings of The 8th ACM Conference on Computer and Communication Security*, 2001.

[10] C. Jutla. Encryption modes with almost free message integrity. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*. Springer-Verlag, May 2001.

[11] H. Krawczyk. The order of encryption and authentication for protecting communication. In *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, Berlin, Germany, 2001. Springer-Verlag.

[12] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[13] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. In *Electronic Notes in Theoretical Computer Science*, volume 32, April 2000.

[14] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In ACM, editor, *ACM Conference on Computer and Communication security*, pages 196–205. ACM Press, 2001.