Encryption Cycles and Two Views of Cryptography

Peeter Laud*

Tartu University, Liivi 2, Tartu, Estonia
Cybernetica AS, Tartu Lab, Lai 6, Tartu, Estonia
peeter@cyber.ee

Abstract. The work by Abadi and Rogaway has started the process of bringing together the two approaches —formal and computational— to cryptography. Their work has also shown, that it is impossible to completely unify these two approaches in their typical forms — there are some principal differences in their security definitions. The difference is in the security of *encryption cycles*. An encryption cycle is a sequence of keys, where each key is encrypted under the next one, and the last key is encrypted under the first one. In formal treatment, they are considered to be secure, but in computational treatment, insecure. In this paper we make the encryption cycles insecure in the formal model (the *Dolev-Yao model*) as well, by slightly strengthening the attacker. For the modified formal model and the classical computational model, the unifying results by Abadi and Rogaway hold unconditionally.

1 Introduction

This work makes a step towards bridging the gap between formal and computational treatments of cryptography in cryptographic protocols. A cryptographic protocol is a sequence of messages between the entities participating in the protocol, together with rules that specify how these messages have to be constructed and which identities they must satisfy. The protocols have been designed with the aim to achieve their security objectives even when there is an adversary present in the system. The adversary can read, intercept and analyse the messages sent by the parties of the protocol, and it can also construct and send new messages.

Two different models for describing, how the adversary can analyse and construct messages, have evolved over the years. In one of them, called the *formal model*, the messages are considered to be formal expressions from a term algebra, the cryptographic (and other) operations are modeled as constructors of terms, and the adversaries abilities are defined by the structure of messages that are known to it. Some of the most prominent examples of this treatment of cryptographic protocols are [9, 8, 1]. In the other model, called the *computational model*

* Supported by Estonian Science Foundation grant #5279

[22, 10, 5], the messages are considered to be bit-strings, the cryptographic operations are modeled as [probabilistic] functions over the set of bit-strings, and the adversary can be any efficient algorithm.

Of these two models, the computational one is arguably more realistic, as messages are encoded as bit-strings in real life. Also, in reality the adversary can be any efficient algorithm. On the other hand, the formal model is easier to reason about, because of its simple algebraic structure. Until very recently, these two models have evolved mostly separately. Only some years ago the first results relating these two models have started to appear. Abadi and Rogaway [3] have shown that if two formal expressions (i.e. messages in the formal model) "look the same" for an adversary, then the [distributions of] bit-strings corresponding to these two expressions also look the same for the adversaries in the computational model. Abadi and Jürjens [2] and Laud [12, 13] have generalised their results for richer formal languages (instead of the values of formal expressions they are handling outputs of programs).

There is an interesting case that the results of Abadi and Rogaway [3], as well as Abadi and Jürjens [2] and Laud [12] (but not [13]) do not cover. Namely, these same-looking formal expressions may not contain *encryption cycles*. The simplest encryption cycle is $\{K\}_K$ — a key whose encryption under itself is made available to the adversary. In the formal model, an adversary that knows just $\{K\}_K$ is unable to obtain K. In the computational model, the encryption primitive is traditionally required to be secure against attacks, whose setup implies that the adversary cannot obtain $\{K\}_K$. One cannot therefore say that if an adversary in the computational model has obtained the encryption of a key under itself, then it cannot find the key itself. Encryption cycles may also be longer, for example, $(\{K_1\}_{K_2}, \{K_2\}_{K_1})$ is an encryption cycle of length 2. Again, they are secure in the formal model but possibly insecure in the computational model.

There are basically two possible approaches to overcome this discrepancy — something has to change in either the formal or the computational model. Changing the computational model involves giving stronger definitions for security of the encryption primitive and constructing primitives satisfying this security definition. This has been done by Black et al. [7]. However, their construction satisfies their security definition only in the *random oracle model* [6]; the existence of random oracles is another security assumption. Changing the formal model involves strengthening the adversary, such that it is able to obtain K from $\{K\}_K$ (and also from longer encryption cycles involving K). Strengthening the adversary is the topic of this paper.

Considering, what is known about the encryption cycles, they should be avoided if possible. The definition of the strengthened adversary, given in this paper, provides a tool for avoiding them.

In this paper we define the strengthened attacker for the formal model. We show that if two formal expressions look the same to this attacker, then the distributions of bit-strings corresponding to these two expressions look the same for the adversaries in the computational model, no matter whether these expressions contain encryption cycles or not. We also show, that if two formal expressions do not contain encryption cycles, then they look the same to the strengthened attacker, if and only if the look the same to the normal attacker.

This paper has the following structure. In Sec. 2 we give a short overview of the work done so far for bringing these two treatments of cryptography together. Sec. 3 introduces the formal model — in Sec. 3.1 we define the language of formal expressions and some syntactic notions, in Sec. 3.2 we recall, how powerful the attacker in its classical shape is, in Sec. 3.3 we describe, how the abilities of the attacker define an equivalence relation over formal expressions, in Sec. 3.4 we introduce the modified attacker. Sec. 4 recalls the computational model, it gives the definition of security of encryption systems in Sec. 4.1 and the translation from formal expressions to (families of probability distributions over) bitstrings in Sec. 4.2. In Sec. 5 we state and prove our main justification for modifying the attacker in the way that we did it in Sec. 3.4 (formal equivalence implies indistinguishability of interpretations). In Sec. 6 we show that for expressions without encryption cycles, the new attacker is no stronger than the classical one. Finally, Sec. 7 concludes.

2 Related Work

We have already mentioned [3, 2, 12]. These works can be considered similar in the sense, that they all attempt to show that some equivalence over a certain class of formally defined objects translates to the indistinguishablity of the computational interpretations of these objects.

A separate approach is that of Pfitzmann et al. [18–20] and Backes [4]. They have devised a framework to faithfully abstract the cryptographic primitives, such that the proofs about protocols using these abstractions would also hold if the abstractions are replaced with actual primitives. This framework is more suitable for abstracting integrity properties than confidentiality properties (which include the indistinguishability of interpretations of expressions). Their abstractions are generally more complex and elaborate than formal expressions and process algebras using them.

Recently, Guttman et al. [11] have related two treatments of cryptography for a certain class of authentication protocols. One of the treatments is the formal treatment, where the messages are formal expressions, the other is based on bitstrings, but in our opinion it should not be called "computational". Rather, it should be called "statistical" — the security definitions are not based on computational complexity, but on statistical closeness. The relationship of the treatments is the usual one — the correctness of the protocol in the formal model implies its correctness in the statistical model.

Mitchell et al. [14, 15, 17] have not actually related the two aspects of cryptography, but they have devised a computational semantics for π-calculus, which is a process algebra that is often used for the formal treatment of cryptographic protocols. They have given correctness proofs for some cryptographic protocols, based on their semantics and corresponding (computational) security definitions.

The proofs are computational in kind, though. There is no translation of proofs from formal to computational world.

3 Formal Model

In the formal model, messages are considered to be formal expressions — elements of a (free) term algebra. Their construction reflects, how they have been put together from simpler messages. The set **Exp** of all formal expressions is defined inductively.

3.1 Formal Expressions

Let **Keys** and **Consts** be two sets, we assume that their intersection is empty. The elements of the set **Keys** represent keys in the formal expressions. The elements of **Consts** represent constants, for example booleans or integers. This representation is really only informal, formally the elements of **Keys** and **Consts** have no further structure.

Definition 1. *The set* **Exp** *of* formal expressions *is the smallest set containing the following elements:*

- *If* $K \in \mathbf{Keys}$*, then* $K \in \mathbf{Exp}$*. If* $C \in \mathbf{Consts}$*, then* $C \in \mathbf{Exp}$*.*
- *If* $E_1, E_2 \in \mathbf{Exp}$*, then the term* (E_1, E_2) *is also an element of* **Exp***.*
- *If* $E \in \mathbf{Exp}$ *and* $K \in \mathbf{Keys}$*, then the term* $\{E\}_K$ *is also an element of* **Exp***.*

The set **Exp** is the "smallest useful" set for describing the exchanged messages in cryptographic protocols. Of course, only protocols that use symmetric encryption as their only cryptographic primitive, can be described. The set **Exp** can be easily extended to include more cryptographic primitives. However, they would only complicate our arguments.

Note that although the elements of **Keys** are intended to model cryptographic keys, they are formally just objects without any further structure. Similarly, the term $\{E\}_K$ is meant to model the encryption of the message modelled by E by the key modelled by K. Formally, however, it has no further structure than being a term, consisting of a binary constructor (denoted $\{\cdot\}_{(\cdot)}$) and two immediate subterms.

As the next step we are going to define, when a formal expression has encryption cycles. This property is defined by the structure of the formal expression. We start with some definitions of simple properties of the structure of formal expressions.

Definition 2. *Let* $E, E' \in \mathbf{Exp}$*. Expression* E' *is a* subexpression *of* E *(denoted* $E' \sqsubseteq E$*), if one of the following holds:*

- $E' = E$*;*
- $E = (E_1, E_2)$ *and either* $E' \sqsubseteq E_1$ *or* $E' \sqsubseteq E_2$*;*
- $E = \{E''\}_K$ *and* $E' \sqsubseteq E''$*.*

We see that K is generally not a subexpression of $\{E\}_K$. We define a second notion, specifically for keys, that also includes the keys that are used for encryption.

Definition 3. *Let $K \in \mathbf{Keys}$ and $E \in \mathbf{Exp}$. We say that K* occurs *in E, if one of the following holds:*

- *$K = E$;*
- *$E = (E_1, E_2)$ and either K occurs in E_1 or K occurs in E_2;*
- *$E = \{E'\}_{K'}$ and either $K = K'$ or K occurs in E'.*

Let $Keys(E) = \{K \in \mathbf{Keys} : K \text{ occurs in } E\}$.

Each formal expression E defines a binary relation "encrypts" on the set of keys occuring in E. Namely, let K, K' be two keys occuring in E, we say that K *encrypts* K', if there exists an expression $E' \in \mathbf{Exp}$, such that $K' \sqsubseteq E'$ and $\{E'\}_K \sqsubseteq E$. For example, K_1 encrypts K_3 in the expression $\{\{(K_3, K_4)\}_{K_2}\}_{K_1}$. We say that E *has encryption cycles*, if the relation "encrypts" is cyclic.

3.2 The Attacker (Classical)

As next we are going to describe the equivalence relation over formal expressions given by Abadi and Rogaway [3]. We already mentioned that two formal expressions are equivalent, if they "look the same" to the attacker. "Looking the same" is given by defining, how each expression "looks". This in turn is given by defining, which subexpressions the attacker can see in an expression.

An *entailment relation* $\vdash \subseteq \mathbf{Exp} \times \mathbf{Exp}$ is given. Intuitively, $E \vdash E'$ means that the attacker can compute the expression E' from the expression E. It is the least relation with the following properties:

- $E \vdash C$ for all $C \in \mathbf{Consts}$;
- $E \vdash E$;
- if $E \vdash (E_1, E_2)$, then $E \vdash E_1$ and $E \vdash E_2$;
- if $E \vdash \{E'\}_K$ and $E \vdash K$, then $E \vdash E'$.

This definition models, what an attacker can obtain from an expression E without any prior knowledge of other keys or expressions. This model of the attacker is ubiquituous in the formal treatment of cryptography.

Example 1. Consider the term $(\{\{K_1\}_{K_2}\}_{K_3}, K_3)$. The attacker can obtain from this term the key K_3, as well as the subterm $\{\{K_1\}_{K_2}\}_{K_3}$. By using the key on that subterm, the attacker can also obtain $\{K_1\}_{K_2}$. But the attacker cannot obtain the keys K_2 or K_1.

3.3 Equivalence Relation

The next step is to define, how a formal expression "looks" to the attacker. The "look" of the expression E is an element of the set of *patterns* (or extended expressions) **Pat**. It is the smallest set that contains the following elements:

– If $K \in \mathbf{Keys}$, then $K \in \mathbf{Pat}$. If $C \in \mathbf{Consts}$, then $C \in \mathbf{Pat}$.
– If $P_1, P_2 \in \mathbf{Pat}$, then the term (P_1, P_2) is also an element of $\mathbf{Pat}$.
– If $P \in \mathbf{Pat}$ and $K \in \mathbf{Keys}$, then the term $\{P\}_K$ is also an element of $\mathbf{Pat}$.
– An object denoted by $\Box$ is an element of $\mathbf{Pat}$.

Intuitively, $\Box$ denotes a term $\{E\}_K$, where the attacker does not know the key K and therefore cannot see E. An element $P \in \mathbf{Pat}$ denotes some expression that may have ciphertexts that the attacker cannot decrypt as subexpressions.

If the attacker knows the keys in the set $\mathbf{K} \subseteq \mathbf{Keys}$, then the formal expression $E \in \mathbf{Exp}$ "looks" to it like $p(E, \mathbf{K}) \in \mathbf{Pat}$, which is defined inductively over the structure of E as follows:

$$
\begin{aligned}
p(K, \mathbf{K}) &:= K \qquad \text{(where } K \in \mathbf{Keys}\text{)} \\
p(C, \mathbf{K}) &:= C \qquad \text{(where } C \in \mathbf{Consts}\text{)} \\
p((E_1, E_2), \mathbf{K}) &:= (p(E_1, \mathbf{K}), p(E_2, \mathbf{K})) \\
p(\{E\}_K, \mathbf{K}) &:= \begin{cases} \{p(E, \mathbf{K})\}_K, & \text{if } K \in \mathbf{K} \\ \Box, & \text{if } K \notin \mathbf{K} \end{cases} .
\end{aligned}
$$

This assumes that the attacker does not obtain any further keys from E. The most natural value for $\mathbf{K}$ is the set of all keys that can be obtained from E. We therefore define

$$
\mathit{pattern}(E) := p(E, \{K \in \mathbf{Keys} : E \vdash K\}) .
$$

The pattern $\mathit{pattern}(E)$ defines, how an expression E "looks" to the attacker. For example, $\mathit{pattern}((\{\{K_1\}_{K_2}\}_{K_3}, K_3)) = (\{\Box\}_{K_3}, K_3)$.

Finally, Abadi and Rogaway [3] define, that two expressions E_1 and E_2 are *equivalent* (denoted $E_1 \equiv E_2$), if $\mathit{pattern}(E_1)$ and $\mathit{pattern}(E_2)$ are equal, and they are *equivalent up to renaming* (denoted $E_1 \cong E_2$), if there exists a bijection σ over the set $\mathbf{Keys}$, such that E_1 is equivalent to $E_2\sigma$, i.e. σ is used to rename all the keys occuring in E_2. The equivalence $\cong$ is more natural — the elements of $\mathbf{Keys}$ should be considered as bound names, they may be subjected to α-conversion. The theorem relating two aspects of cryptography also holds for $\cong$ — if $E_1 \cong E_2$ and E_1 and E_2 have no encryption cycles, then the computational interpretations of E_1 and E_2 are indistinguishable, as shown in [3].

3.4 The Attacker (Our Definition)

Strengthening the attacker involves changing the entailment relation $\vdash$. After giving a new definition for $\vdash$, the equivalence of formal expressions can be defined exactly as in Sec. 3.3.

In fact, we are not going to give just a single relation $\vdash$, but a family of relations $\vdash_{\mathbf{K}}$, parametrised by $\mathbf{K} \subseteq \mathbf{Keys}$. Intuitively, $E \vdash_{\mathbf{K}} E'$ means that the attacker can compute E' from E, if it has access to oracles that decrypt with K, where $K \in \mathbf{K}$. The relations $\vdash_{\mathbf{K}}$ are defined as follows:

(i). $E \vdash_\emptyset C$ for all $C \in \mathbf{Consts}$;
(ii). $E \vdash_\emptyset E$;
(iii). if $E \vdash_{\mathbf{K}} E'$ and $\mathbf{K} \subseteq \mathbf{K}'$, then $E \vdash_{\mathbf{K}'} E'$;
(iv). if $E \vdash_{\mathbf{K}} (E_1, E_2)$, then $E \vdash_{\mathbf{K}} E_1$ and $E \vdash_{\mathbf{K}} E_2$;
(v). if $E \vdash_{\mathbf{K}} \{E'\}_K$, then $E \vdash_{\mathbf{K}\cup\{K\}} E'$;
(vi). if $E \vdash_{\mathbf{K}\cup\{K\}} E'$ and $E \vdash_{\mathbf{K}} K$, then $E \vdash_{\mathbf{K}} E'$;
(vii). if $E \vdash_{\mathbf{K}\cup\{K\}} K$, then $E \vdash_{\mathbf{K}} K$.

In this definition, the items (i), (ii) and (iv) are the same as in the definition of $\vdash$. The item (iii) is an obvious monotonicity property. The item (v) is a bit more optimistic (from the attacker's point of view) than the corresponding item in the definition of $\vdash$ — one does not need the key K to decrypt a ciphertext $\{E'\}_K$, it is enough to have access to an oracle that decrypts with K. The item (vi) says that one way to obtain that access is to have K itself. The item (vii) breaks encryption cycles. It says that if a key K is obtainable by using an oracle that decrypts with K, then this key is (or may be) also obtainable without that oracle. Let us see some examples (with encryption cycles in them).

Example 2. The expression E is $\{K\}_K$.

(1). $E \vdash_{\{K\}} K$ is derived by items (ii) and (v);
(2). $E \vdash_\emptyset K$ is derived from (1) by item (vii).

Example 3. The expression E is $(\{K_1\}_{K_2}, \{K_2\}_{K_1})$.

(1). $E \vdash_{\{K_2\}} K_1$ is derived by items (ii), (iv) and (v);
(2). $E \vdash_{\{K_1\}} K_2$ is derived similarly;
(3). $E \vdash_{\{K_1,K_2\}} K_1$ is derived from (1) by item (iii);
(4). $E \vdash_{\{K_1\}} K_1$ is derived from (3) and (2) by item (vi);
(5). $E \vdash_\emptyset K_1$ is derived from (4) by item (vii).

Example 4. The expression E is $\{\{(K_1, K_2)\}_{K_1}\}_{K_2}$.

(1). $E \vdash_{\{K_1,K_2\}} (K_1, K_2)$ is derived by items (ii) and (v);
(2). $E \vdash_{\{K_1,K_2\}} K_1$ is derived from (1) by item (iv);
(3). $E \vdash_{\{K_2\}} K_1$ is derived from (2) by item (vii);
(4). $E \vdash_{\{K_2\}} (K_1, K_2)$ is derived from (1) and (3) by item (vi);
(5). $E \vdash_{\{K_2\}} K_2$ is derived from (4) by item (iv);
(6). $E \vdash_\emptyset K_2$ is derived from (5) by item (vii);
(7). $E \vdash_\emptyset K_1$ is derived from (3) and (6) by item (vi).

We again define, how a formal expression "looks" to an attacker. Now we use the relation $\vdash_\emptyset$ instead of the relation $\vdash$. Let

$$\mathit{pattern}^{\mathrm{EC}}(E) := p(E, \{K \in \mathbf{Keys} : E \vdash_\emptyset K\}) \ .$$

Similarly, we define

$$\begin{aligned} E_1 &\equiv^{\mathrm{EC}} E_2 && \text{if } \mathit{pattern}^{\mathrm{EC}}(E_1) = \mathit{pattern}^{\mathrm{EC}}(E_2) \\ E_1 &\cong^{\mathrm{EC}} E_2 && \text{if } \exists \text{ bij. } \sigma \text{ over } \mathbf{Keys} : E_1 \equiv^{\mathrm{EC}} E_2\sigma \ . \end{aligned}$$

In Sec. 5 we will show that if $E_1 \cong^{\mathrm{EC}} E_2$, then the computational interpretations of E_1 and E_2 are indistinguishable.

4 Computational Model

In this section we give an overview, what constitutes a symmetric encryption system and what does its security mean. We also relate two models, by associating a family of distributions over bit-strings to each formal expression. The content of this section is not original, it repeats [3, Sec. 4 and Sec. 5.1].

Some general concepts have to be introduced first. Let $\mathsf{String} = \{0,1\}^*$ denote the set of all bit-strings. A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible*, if for all polynomials p over $\mathbb{R}$, $|\epsilon(n)|$ is asymptotically smaller that $|1/p(n)|$. Let D_n be a probability distribution over String. We write $x \leftarrow D_n$ to denote that the variable x is sampled according to D_n. Writing $x, y \leftarrow D_n$ means that the variables x and y are sampled according to D_n *independently of each other*. Let $D = \{D_n\}_{n\in\mathbb{N}}$ and $D' = \{D'_n\}_{n\in\mathbb{N}}$ be two families of probability distributions over String. We say that D and D' are *indistinguishable* (denoted $D \approx D'$), if for every probabilistic algorithm $\mathcal{A}$ that works in polynomial time in its first argument, the difference of probabilities

$$\Pr[\mathcal{A}(n,x) = 1 \,:\, x \leftarrow D_n] - \Pr[\mathcal{A}(n,x) = 1 \,:\, x \leftarrow D'_n] \tag{1}$$

is negligible in n. Intuitively, indistinguishability means "being/looking the same" in the computational model.

A simple property of indistinguishability is the transitivity — if $D \approx D'$ and $D' \approx D''$, then $D \approx D''$. For the proof of this statement, see, for example [13, Sec. 2.2.2].

4.1 Encryption Systems

A symmetric encryption system is a triple of (probabilistic) algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, whose inputs and outputs are bit-strings. The *key generation algorithm* $\mathcal{G}$ takes as its input a *security parameter* $n \in \mathbb{N}$ (somehow encoded as a bit-string) and returns a key k from a suitable domain $\mathsf{Key} \subseteq \mathsf{String}$. The *encryption algorithm* $\mathcal{E}$ takes as its input a key k and a *plaintext* m and produces a *ciphertext* $\mathcal{E}_k(m)$. The *decryption algorithm* $\mathcal{D}$ takes as its input a key k and a ciphertext c and produces the underlying plaintext, such that $\mathcal{D}_k(\mathcal{E}_k(m)) = m$ for all m and k. The running times of the algorithms $\mathcal{G}$, $\mathcal{E}$ and $\mathcal{D}$ must be polynomial in n.

Let $\mathbf{0}$ be a fixed member of String. The encryption system is *type-0 secure*[1] (see [3] for discussions), if for all algorithms $\mathcal{A}$ running in polynomial time of its argument, the difference of probabilities

$$\Pr[\mathcal{A}^{\mathcal{E}_k(\cdot),\mathcal{E}_{k'}(\cdot)}(n) = 1 \,:\, k, k' \leftarrow \mathcal{G}(n)] - \Pr[\mathcal{A}^{\mathcal{E}_k(\mathbf{0}),\mathcal{E}_k(\mathbf{0})}(n) = 1 \,:\, k \leftarrow \mathcal{G}(n)] \tag{2}$$

is negligible (in n).

In the following we assume that $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a fixed type-0 secure encryption system. Abadi and Rogaway [3] have shown, how to construct such systems.

[1] alternative name of this property is: *repetition concealing, which-key concealing and message-length concealing*

algorithm SAMPLE(n, E)
  INITIALISE$(n, Keys(E))$
  return CONVERT(E)

algorithm INITIALISE$(n, \mathbf{K})$
  for all $K \in \mathbf{K}$ **do** $\tau(K) \leftarrow \mathcal{G}(n)$

algorithm CONVERT(E)
  case E **of**
    $K \in$ **Keys**: **return** $\langle \tau(K), \mathsf{key} \rangle$
    $C \in$ **Consts**: **return** $\langle [\![C]\!], \mathsf{const} \rangle$
    (E_1, E_2): **return** $\langle \text{CONVERT}(E_1), \text{CONVERT}(E_2), \mathsf{pair} \rangle$
    $\{E'\}_K$: **return** $\langle \mathcal{E}_{\tau(K)}(\text{CONVERT}(E')), \mathsf{ciphertext} \rangle$

Fig. 1. Algorithm that samples $[\![E]\!]$

4.2 Computational Interpretation of Formal Expressions

Two models of cryptography are related by associating a family of distributions over bit-strings $[\![E]\!]$ to each formal expression E. The formal expression serves here as a "program" that specifies, how the algorithm sampling these distributions works.

Let key, const, pair and $\mathsf{ciphertext}$ be four fixed bit-strings. For bit-strings $x_1, \ldots, x_k$ let $\langle x_1, \ldots, x_k \rangle$ be the encoding of the tuple $(x_1, \ldots, x_k)$ as a bit-string. I.e. $\langle x_1, \ldots, x_k \rangle$ is not just the concatenation $x_1 \cdots x_k$, but it contains enough information to recover $x_1, \ldots, x_k$ from it. In other words, $\langle \cdot \rangle : \mathsf{String}^* \rightarrow \mathsf{String}$ is an injective function. The n-th component of the family of distributions $[\![E]\!]$ is defined by stating, that it is sampled by the algorithm SAMPLE(n, E) in Fig. 1.

This algorithm starts by picking a value for each key K that occurs in the expression E, where the values for different keys are picked independently of each other. The value that is picked for the key K is denoted by $\tau(K)$. After that, the sampling algorithm proceeds recursively over the expression structure. A key K is mapped to $\tau(K)$. A constant $C \in$ **Consts** is mapped to the representation of C as a bit-string $[\![C]\!]$ (i.e. $[\![\cdot]\!]$: **Consts** $\rightarrow \mathsf{String}$ is a mapping that is fixed beforehand). The mappings of (E_1, E_2) and $\{E\}_K$ are created from the mappings of their components in the most natural way. All interpretations of expressions are tagged by the type of their outermost constructor. This tagging ensures that all bit-strings produced by the sampling algorithm can be uniquely decomposed. Unique decomposition makes the interpretations of formal expressions more similar to those expressions themselves, however, it is not necessary for the proof of the next theorem.

5 Equivalence of Models

The equivalence relations over the formal expressions and their computational interpretations are related by the following theorem:

algorithm $\textsc{Sample}(n, P)$
  $\textsc{Initialise}(n, \mathit{Keys}(P))$
  return $\textsc{Convert}(P)$

algorithm $\textsc{Initialise}(n, \mathbf{K})$
  for all $K \in \mathbf{K}$ **do** $\tau(K) \leftarrow \mathcal{G}(n)$
  $\kappa \leftarrow \mathcal{G}(n)$

algorithm $\textsc{Convert}(P)$
  case P **of**
    $K \in \mathbf{Keys}$: **return** $\langle \tau(K), \mathsf{key} \rangle$
    $C \in \mathbf{Consts}$: **return** $\langle [\![C]\!], \mathsf{const} \rangle$
    (P_1, P_2): **return** $\langle \textsc{Convert}(P_1), \textsc{Convert}(P_2), \mathsf{pair} \rangle$
    $\{P'\}_K$: **return** $\langle \mathcal{E}_{\tau(K)}(\textsc{Convert}(P')), \mathsf{ciphertext} \rangle$
    $\Box$: **return** $\langle \mathcal{E}_{\kappa}(\mathbf{0}), \mathsf{ciphertext} \rangle$

Fig. 2. Algorithm that samples $[\![P]\!]$

Theorem 1. *Let $E_1, E_2 \in \mathbf{Exp}$, such that $E_1 \cong^{\mathrm{EC}} E_2$. Then $[\![E_1]\!] \approx [\![E_2]\!]$.*

Proof. The proof is rather similar to that in [3], only the setup of the hybrid argument requires a more elaborate step for sorting the hidden keys of the expressions. The *recoverable* and *hidden* keys of an expression E are defined by

$$\begin{aligned} \mathit{recoverable}(E) &:= \{K \in \mathbf{Keys} \,:\, E \vdash_{\emptyset} K\} \\ \mathit{hidden}(E) &:= \mathit{Keys}(E) \backslash \mathit{recoverable}(E) \ . \end{aligned}$$

First we are going to extend the interpretation of expressions given in Fig. 1 to patterns. Then we are going to show $[\![E]\!] \approx [\![\mathit{pattern}^{\mathrm{EC}}(E)]\!]$ for all $E \in \mathbf{Exp}$. This more or less conlcudes the proof, because obviously $[\![E]\!] = [\![E\sigma]\!]$ for all bijections σ over **Keys**. Also, for concluding the proof we need to use the transitivity property of indistinguishability.

The extension of the computational interpretation to the language of patterns is given by redefining the algorithm Sample as in Fig. 2. We see that there are exactly two changes with respect to Fig. 1. First, the algorithm Initialise generates an extra key κ. Second, the algorithm Convert uses this extra key for interpreting the symbol $\Box$.

The next step in the proof is defining the hybrids for showing that $[\![E]\!]$ and $[\![\mathit{pattern}^{\mathrm{EC}}(E)]\!]$ are indistinguishable. This starts with suitably ordering the keys in $\mathit{hidden}(E)$; the hybrids are defined by successively "opening up" the hidden keys of E. This way, the first hybrid (where all hidden keys are covered) is equal to $\mathit{pattern}^{\mathrm{EC}}(E)$ and the last hybrid (where all hidden keys are visible) is equal to E. In [3], the keys were topologically sorted according to the relation "encrypts"; any order possibly resulting from such sort was suitable. We cannot sort the hidden keys this way, because we allow the relation "encrypts" to have cycles. However, suitable orders still exist.

Let h be the cardinality of $hidden(E)$ and let $\Sigma := \{1,\ldots,h\} \to hidden(E)$. For each $\pi \in \Sigma$ and $i \in \{0,\ldots,h\}$ define

$$\begin{aligned}\mathbf{K}_i^\pi &:= \{\pi(1),\ldots,\pi(i)\}\\ \mathcal{K}_i^\pi &:= \mathbf{K}_{i-1}^\pi \cup \{K \in \mathbf{Keys} : E \vdash_{\mathbf{K}_i^\pi} K\}\end{aligned} \tag{3}$$

and let $\mathcal{K}^\pi := (\mathcal{K}_1^\pi,\ldots,\mathcal{K}_h^\pi)$. We are looking for a $\pi \in \Sigma$, such that

$$\mathcal{K}_i^\pi = \mathbf{K}_{i-1}^\pi \cup recoverable(E) \tag{4}$$

for each $i \in \{1,\ldots,h\}$. Note that (4) is the minimal possible value of $\mathcal{K}_i^\pi$.

Such π exists. To show it, we need to define two more notions. First, define a partial order on the set $\{\mathcal{K}^\pi : \pi \in \Sigma\}$. The order is defined componentwise, the components are ordered by set inclusion. Second, for each $\pi \in \Sigma$ and $i \in \{1,\ldots,h\}$ define $\mathsf{fo}_i^\pi \in \{1,\ldots,h\} \cup \{\infty\}$ by

$$\mathsf{fo}_i^\pi = \min\{j \in \{1,\ldots,h\} : \pi(i) \in \mathcal{K}_j^\pi\} \ .$$

An important property of fo_i^π is $\mathsf{fo}_i^\pi \neq i$. Indeed, suppose that there exist $\pi \in \Sigma$ and $i \in \{1,\ldots,h\}$, such that $\mathsf{fo}_i^\pi = i$. We have $\pi(i) \in \mathcal{K}_i^\pi$. From (3) follows $E \vdash_{\mathbf{K}_i^\pi} \pi(i)$, because $\pi(i) \notin \mathbf{K}_{i-1}^\pi$. From item (vii) of the definition of $\vdash_{\mathbf{K}}$ follows $E \vdash_{\mathbf{K}_{i-1}^\pi} \pi(i)$. Therefore $\pi(i) \in \mathcal{K}_{i-1}^\pi$ (if $i > 1$) or $\pi(i) \in recoverable(E)$. A contradiction with the definition of fo_i^π.

For any $\pi \in \Sigma$, where $\mathcal{K}^\pi$ is minimal, the condition (4) is satisfied. Indeed, suppose that $\mathcal{K}^\pi$ is minimal, but (4) does not hold for some i. Let i be the smallest index, such that (4) does not hold. There exists $j \in \{i+1,\ldots,h\}$, such that $\pi(j) \in \mathcal{K}_i^\pi$; we have $E \vdash_{\mathbf{K}_i^\pi} \pi(j)$ and $\mathsf{fo}_j^\pi = i$. Construct $\pi' \in \Sigma$ as follows:

$$\pi' = \left\{\begin{array}{ccccccccc} 1 & & i-1 & i & i+1 & & j & j+1 & & h \\ \downarrow & \cdots & \downarrow & \downarrow & \downarrow & \cdots & \downarrow & \downarrow & \cdots & \downarrow \\ \pi(1) & & \pi(i-1) & \pi(j) & \pi(i) & & \pi(j-1) & \pi(j+1) & & \pi(h) \end{array}\right\} \ .$$

For $l \in \{1,\ldots,i-1,j+1,\ldots,h\}$ we have $\mathcal{K}_l^{\pi'} = \mathcal{K}_l^\pi$. For $l \in \{i+1,\ldots,j\}$ we have

$$\begin{aligned}\{K \in \mathbf{Keys} : E \vdash_{\mathbf{K}_l^{\pi'}} K\} \subseteq \{K \in \mathbf{Keys} : E \vdash_{\mathbf{K}_{l-1}^\pi \cup \pi(j)} K\} = \\ \{K \in \mathbf{Keys} : E \vdash_{\mathbf{K}_{l-1}^\pi} K\},\end{aligned}$$

where the equality holds because of $E \vdash_{\mathbf{K}_{l-1}^\pi} \pi(j)$. Therefore $\mathcal{K}_l^{\pi'} = \mathcal{K}_{l-1}^\pi \subseteq \mathcal{K}_l^\pi$ for all $l \in \{i+1,\ldots,j\}$. Also, we have $\mathcal{K}_i^{\pi'} \subseteq \mathcal{K}_{i+1}^{\pi'} = \mathcal{K}_i^\pi$. We have shown that $\mathcal{K}^{\pi'} \leq \mathcal{K}^\pi$.

We also have $\mathcal{K}_i^{\pi'} \neq \mathcal{K}_i^\pi$. Namely, we have $\pi'(i) \notin \mathcal{K}_l^{\pi'}$ for $l \in \{1,\ldots,i-1\}$. This follows from the construction of π' and from the definition of i as the smallest index for which (4) is not satisfied. We also have $\pi'(i) \notin \mathcal{K}_i^{\pi'}$, because $\mathsf{fo}_i^{\pi'}$ cannot be equal to i. On the other hand, we have $\pi'(i) = \pi(j) \in \mathcal{K}_i^\pi$. Therefore $\mathcal{K}^{\pi'} < \mathcal{K}^\pi$, this contradicts the minimality of $\mathcal{K}^\pi$.

```
algorithm A_0^{f(·),g(·)}(n)
    Initialise(n, K^π_{i−1} ∪ recoverable(E))
    return A(n, Convert'(E))

algorithm Initialise(n, K)
    for all K ∈ K do τ(K) ← G(n)

algorithm Convert'(E)
    case E of
        K ∈ Keys:    return ⟨τ(K), key⟩
        C ∈ Consts:  return ⟨[[C]], const⟩
        (E_1, E_2):  return ⟨Convert'(E_1), Convert'(E_2), pair⟩
        {E'}_K:      if K ∈ K^π_{i−1} then
                         return ⟨E_{τ(K)}(Convert'(E')), ciphertext⟩
                     else if K = π(i) then
                         return ⟨f(Convert'(E')), ciphertext⟩
                     else
                         return ⟨g(0), ciphertext⟩
```

Fig. 3. Algorithm that breaks the type-0 security of the encryption system

Let $\pi \in \Sigma$ be such, that $\mathcal{K}^\pi$ is minimal. This fixes the order on the hidden keys; the rest of the proof is more or less the same as in [3]. We define the patterns $P_0, \ldots, P_h \in \mathbf{Pat}$ by $P_i := p(E, \mathbf{K}_i^\pi)$. Obviously $P_0 = \mathit{pattern}^{\mathrm{EC}}(E)$ and $P_h = E$.

We claim that for each $i \in \{1, \ldots, h\}$, the interpretations $[\![P_{i-1}]\!]$ and $[\![P_i]\!]$ are indistinguishable. Indeed, suppose that $\mathcal{A}$ is an algorithm, such that (1) is nonnegligible for the families of distributions $[\![P_{i-1}]\!]$ and $[\![P_i]\!]$. Consider the algorithm $\mathcal{A}_0^{(\cdot),(\cdot)}$ in Fig. 3. We claim that if its oracle f is $\mathcal{E}_k(\cdot)$ and its oracle g is $\mathcal{E}_{k'}(\cdot)$, then the second argument given to $\mathcal{A}$ is distributed according to $[\![P_i]\!]_n$. If the oracles f and g are both $\mathcal{E}_k(\mathbf{0})$, then the second argument given to $\mathcal{A}$ is distributed according to $[\![P_{i-1}]\!]_n$.

First we argue that the algorithm $\mathcal{A}_0^{(\cdot),(\cdot)}$ never reaches a state where its further action is undefined. This may happen, if the variant $K \in \mathbf{Keys}$ is chosen in the algorithm Convert' and $\tau(K)$ is undefined (note that Initialise does not necessarily define τ for all keys that occur in E). However, it is easy to see (follows from the definitions of $p(\cdot,\cdot)$ and $\vdash_{\mathbf{K}}$) that the pattern P_i only contains keys in $\mathcal{K}_i^\pi$. The property (4) guarantees that we have all these keys available.

When f is $\mathcal{E}_k(\cdot)$ and g is $\mathcal{E}_{k'}(\cdot)$, then the key that f uses corresponds to the key $\tau(\pi(i))$ in Fig. 2, and the key that g uses corresponds to the key κ in Fig. 2. When both f and g are $\mathcal{E}_k(\mathbf{0})$, then the key that they both use corresponds to the key κ in Fig. 2.

By our assumption, the algorithm $\mathcal{A}$ can distinguish $[\![P_{i-1}]\!]$ and $[\![P_i]\!]$, therefore the difference (2) is not negligible for the algorithm $\mathcal{A}_0$. But this contradicts our choice of the encryption system as a type-0 secure system. Hence

$[\![P_{i-1}]\!] \approx [\![P_i]\!]$ and by the transitivity of indistinguishability, $[\![\mathit{pattern}^{\mathrm{EC}}(E)]\!] = [\![P_0]\!] \approx [\![P_h]\!] = [\![E]\!]$. □

This gives the justification for our definition of $\vdash_{\mathbf{K}}$.

6 On the Added Strength of the Attacker

We have defined an equivalence relation $\cong^{\mathrm{EC}}$ over the formal expressions. Let us compare it to the relation $\cong$ defined in [3]. Obviously, it makes sense to only consider expressions that do not contain encryption cycles. The following result holds, showing that the relation $\cong^{\mathrm{EC}}$ is as precise as $\cong$.

Proposition 1. *Let $E_1, E_2 \in \mathbf{Exp}$, such that E_1 and E_2 have no encryption cycles. Then $E_1 \cong E_2$ iff $E_1 \cong^{\mathrm{EC}} E_2$.*

Proof. It is enough to show that if $E \in \mathbf{Exp}$ has no encryption cycles, then for all $E' \in \mathbf{Exp}$, $E \vdash E'$ holds iff $E \vdash_{\emptyset} E'$ holds. Indeed, from this equality of $\vdash$ and $\vdash_{\emptyset}$ follows, that $\mathit{pattern}(E) = \mathit{pattern}^{\mathrm{EC}}(E)$. The direction "if $E \vdash E'$ holds then $E \vdash_{\emptyset} E'$ holds" is trivial. Consider the other direction.

The claim "if $E \vdash_{\emptyset} E'$ holds then $E \vdash E'$ holds" is proved in two steps. The first step is to show that if $E \vdash_{\mathbf{K}} E'$ holds, then it can be derived without using item (vii) in the definition of $\vdash_{\mathbf{K}}$.

Let T be a derivation tree for $E \vdash_{\mathbf{K}} K$, i.e. its root is labeled with $E \vdash_{\mathbf{K}} K$, its vertices are labeled with terms of the form $E \vdash_{\mathbf{K}'} E'$ for some $E' \in \mathbf{Exp}$ and $\mathbf{K}' \subseteq \mathbf{Keys}$, and its edges correspond to the items in the definition of $\vdash_{\mathbf{K}}$. Note that $K \sqsubseteq E'$ holds for all expressions E' occuring in the vertices of T. We show how to construct a derivation tree T' for $E \vdash_{\mathbf{K}\backslash\{K\}} K$, such that for all $E \vdash_{\mathbf{K}'} E'$ that are labels of vertices of T', $K \notin \mathbf{K}'$ holds. In the tree T', item (vii) of the definition of $\vdash_{\mathbf{K}}$ cannot have been used for the key K. By repeatedly using this construction, we can remove all occurrences of item (vii) of the definition of $\vdash_{\mathbf{K}}$ from the derivation trees.

Let $E \vdash_{\mathbf{K}'} E'$ be the label of some vertex v of the tree T and let $T_{\mathbf{K}',E'}$ be the subtree of T that is rooted in this vertex. The tree $T'_{\mathbf{K}',E'}$, whose root is labeled with $E \vdash_{\mathbf{K}'\backslash\{K\}} E'$ can be constructed as follows.

Let the immediate descendants of v be v_i, where $i \in \{1, \ldots, \deg v\}$. Let v_i be labeled by $E \vdash_{\mathbf{K}_i} E_i$. First construct the trees $T'_{\mathbf{K}_i,E_i}$ whose root vertices are labeled by $E \vdash_{\mathbf{K}_i\backslash\{K\}} E_i$. The construction of $T'_{\mathbf{K}',E'}$ depends on the item in the definition of $\vdash_{\mathbf{K}}$ that was used to derive $E \vdash_{\mathbf{K}'} E'$. In most cases we have

$$\Big(\bigwedge_{i=1}^{\deg v} E \vdash_{\mathbf{K}_i} E_i \Rightarrow E \vdash_{\mathbf{K}'} E'\Big) \Rightarrow \Big(\bigwedge_{i=1}^{\deg v} E \vdash_{\mathbf{K}_i\backslash\{K\}} E_i \Rightarrow E \vdash_{\mathbf{K}'\backslash\{K\}} E'\Big) \quad (5)$$

and the tree $T'_{\mathbf{K}',E'}$ can be constructed by taking a new vertex v', labeling it with $E \vdash_{\mathbf{K}'\backslash\{K\}} E'$, and letting its immediate descendants be the root vertices of $T'_{\mathbf{K}_i,E_i}$. The implication (5) does not hold in the following cases:

– $E \vdash_{\mathbf{K}'} E'$ is derived by item (v) in the definition of $\vdash_{\mathbf{K}}$ and $E_1 = \{E'\}_K$. This case is impossible, because $K \sqsubseteq E'$ and therefore we have an encryption cycle in E.
– $E \vdash_{\mathbf{K}'} E'$ is derived by item (vi) in the definition of $\vdash_{\mathbf{K}}$, $E_1 = E'$ and $\mathbf{K}_1 = \mathbf{K}' \cup \{K\}$. In this case we just take $T'_{\mathbf{K}',E'} = T'_{\mathbf{K}_1,E_1}$.
– $E \vdash_{\mathbf{K}'} E'$ is derived by item (vii) in the definition of $\vdash_{\mathbf{K}}$. We again take $T'_{\mathbf{K}',E'} = T'_{\mathbf{K}_1,E_1}$.

The second step for proving the claim "if $E \vdash_{\emptyset} E'$ holds then $E \vdash E'$ holds" is to show, that if $E \vdash_{\mathbf{K}} E'$ holds (we assume that it has been derived without using item (vii) in the definition of $\vdash_{\mathbf{K}}$), then there exists a derivation tree for $E \vdash E'$, whose leaves can be labeled not only with $E \vdash C$ or $E \vdash E$, but also with $E \vdash K$ for $K \in \mathbf{K}$. Consider, by which item has $E \vdash_{\mathbf{K}} E'$ been derived.

(i). The derivation tree of $E \vdash C$ has one vertex, labeled with $E \vdash C$.
(ii). The derivation tree of $E \vdash E$ has one vertex, labeled with $E \vdash E$.
(iii). The derivation tree for $E \vdash E'$ that may have $E \vdash K$, where $K \in \mathbf{K}'$, as additional axioms, has already been constructed.
(iv). The derivation tree for $E \vdash E_i$ with the additional axioms $E \vdash K$, where $K \in \mathbf{K}$, consists of a root vertex labeled with $E \vdash E_i$, whose descendant is the root vertex of the derivation tree for $E \vdash (E_1, E_2)$ with the additional axioms $E \vdash K$, where $K \in \mathbf{K}$.
(v). The derivation tree for $E \vdash E'$ with the additional axioms $E \vdash K'$, where $K' \in \mathbf{K} \cup \{K\}$, consists of a root vertex labeled with $E \vdash E'$ and having two descendants. The first descendant is the root vertex of the derivation tree for $E \vdash \{E'\}_K$ with the additional axioms $E \vdash K'$, where $K' \in \mathbf{K}$. The second descendant is a vertex labeled with $E \vdash K$ having no descendants.
(vi). The derivation tree for $E \vdash E'$ with the additional axioms $E \vdash K'$, where $K' \in \mathbf{K}$, consists of the derivation tree for $E \vdash E'$ with the additional axioms $E \vdash K'$, where $K' \in \mathbf{K} \cup \{K\}$, that is modified by replacing the axioms $E \vdash K$ with the derivation trees for $E \vdash K$ with the additional axioms $E \vdash K'$, where $K' \in \mathbf{K}$.

We have shown that if $E \vdash_{\emptyset} E'$ holds then there exists a derivation tree for $E \vdash E'$. □

7 Conclusions

We have amended the Dolev-Yao model, making it in some sense more realistic. By "more realistic" we mean closer to the computational model of cryptography, which can be considered to be a good model of the real world. Our amendment does not make the model much more complex, it should still be suitable for (mechanical) analysis.

Micciancio and Warinschi [16] have refined the formal equivalence $\cong$, such that the computational interpretations of equivalent formal messages are indistinguishable also if the used encryption system is only *type-1* secure (compared to type-0 security, message length may be revealed). They change the language of patterns, by replacing the undecryptable ciphertext $\Box$ with a collection $\Box_{\ell_1}, \Box_{\ell_2}, \ldots$, denoting the undecryptable ciphertexts of length ℓ_1, ℓ_2, etc. Obviously, the equivalence $\cong^{\mathrm{EC}}$ can be similarly refined.

References

1. Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, January 1999.
2. Martín Abadi and Jan Jürjens. Formal Eavesdropping and Its Computational Interpretation. In Naoki Kobayashi and Benjamin C. Pierce, editors, *Theoretical Aspects of Computer Software, 4th International Symposium, TACS 2001*, volume 2215 of *LNCS*, pages 82–94, Sendai, Japan, September 2001. Springer-Verlag.
3. Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *International Conference IFIP TCS 2000*, volume 1872 of *LNCS*, pages 3–22, Sendai, Japan, August 2000. Springer-Verlag.
4. Michael Backes. *Cryptographically Sound Analysis of Security Protocols.* PhD thesis, Universität des Saarlandes, 2002.
5. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 1997. IEEE Computer Society Press.
6. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, November 1993. ACM Press.
7. John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption Scheme Security in the Presence of Key-Dependent Messages. In *Proceedings of the Ninth Annual Workshop in Selected Areas in Cryptography*, St John's, Newfoundland, August 2002.
8. Michael Burrows, Martín Abadi, and Roger M. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.
9. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, March 1983.
10. Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
11. Joshua D. Guttman, F. Javier Thayer, and Lenore D. Zuck. The faithfulness of abstract protocol analysis: message authentication. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 186–195, Philadelphia, PA, November 2001. ACM Press.
12. Peeter Laud. Semantics and Program Analysis of Computationally Secure Information Flow. In Sands [21], pages 77–91.
13. Peeter Laud. *Computationally Secure Information Flow.* PhD thesis, Universität des Saarlandes, 2002.
14. Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A Probabilistic Poly-Time Framework for Protocol Analysis. In *CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 112–121, San Francisco, California, November 1998. ACM Press.

15. Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. Probabilistic Polynomial-Time Equivalence and Security Analysis. In Jeanette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems*, volume 1708 of *LNCS*, pages 776–793, Toulouse, France, September 1999. Springer-Verlag.
16. Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. In *Workshop on Issues in the Theory of Security - WITS 2002*, Portland, Oregon, January 2002.
17. John C. Mitchell. Probabilistic Polynomial-Time Process Calculus and Security Protocol Analysis. In Sands [21], pages 23–29.
18. Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Cryptographic Security of Reactive Systems. In Steve Schneider and Peter Ryan, editors, *Workshop on Secure Architectures and Information Flow*, volume 32 of *Electronic Notes in Theoretical Computer Science*, Royal Holloway, University of London, 2000. Elsevier Science.
19. Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *CCS 2000, Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 245–254, Athens, Greece, November 2000. ACM Press.
20. Birgit Pfitzmann and Michael Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *2001 IEEE Symposium on Security and Privacy*, pages 184–200, Oakland, California, May 2001. IEEE Computer Society Press.
21. David Sands, editor. *Programming Languages and Systems, 10th European Symposium on Programming, ESOP 2001*, volume 2028 of *LNCS*, Genova, Italy, April 2001. Springer-Verlag.
22. Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, November 1982. IEEE Computer Society Press.