An abridged, preliminary version of this paper will appear in the proceedings of the *IFIP International Conference on Theoretical Computer Science* (IFIP TCS2000), Sendai, Japan, August 2000. This is the full paper.

Reconciling Two Views of Cryptography

(The Computational Soundness of Formal Encryption)

Martín Abadi
Bell Labs Research
Lucent Technologies
abadi@lucent.com
www.pa.bell-labs.com/~abadi

Phillip Rogaway
Department of Computer Science
University of California at Davis
rogaway@cs.ucdavis.edu
www.cs.ucdavis.edu/~rogaway

January 31, 2001

Abstract

Two distinct, rigorous views of cryptography have developed over the years, in two mostly separate communities. One of the views relies on a simple but effective formal approach; the other, on a detailed computational model that considers issues of complexity and probability. There is an uncomfortable and interesting gap between these two approaches to cryptography. This paper starts to bridge the gap, by providing a computational justification for a formal treatment of encryption.

Contents

1 Two Views of Cryptography

A fairly abstract view of cryptographic operations is often adequate for the design, analysis, and implementation of systems that use cryptography. For example, it is often convenient to ignore the details of an encryption function, and to work instead with a high-level description of what encryption is supposed to achieve.

At least two distinct abstract views of cryptographic operations have developed over the years. They are both consistent and they have both been useful, but they come from two mostly separate communities and they are quite different. In one of them, cryptographic operations are seen as functions on a space of symbolic (formal) expressions; their security properties are also modeled formally (e.g., [5, 13, 16–18, 28–30, 32, 34–37, 41]).[1] In the other, cryptographic operations are seen as functions on strings of bits; their security properties are defined in terms of the probability and computational complexity of successful attacks (e.g., [7–9, 11, 22–26, 44]).

There is an uncomfortable gap between these two views. In this paper, we call attention to this gap and start to bridge it. Representing the two views, we give two accounts of symmetric (shared-key) encryption: a simple one, based on a formal system, and a more elaborate one, based on a computational model. Our main theorem is a soundness result that relates the two accounts. It establishes that secrecy properties that can be proved in the formal world are true in the computational world. Thus, we obtain a computational justification for the formal treatment of encryption.

As we relate the two accounts of encryption, we identify and make explicit some important choices. In particular, our main theorem excludes certain encryption cycles (such as encrypting a key with itself). A restriction along these lines is essential within the prevailing computational approach; in contrast, formal methods typically ignore cycles. We also consider, for example, whether two ciphertexts may manifest whether they were produced using the same key.

We believe that this paper suggests a profitable line of further research. It will take a significant research effort to relate the views of the people who invent, implement, break, and use cryptography. Continuing this work, it would be worthwhile to consider other cryptographic operations (such as signatures and hash functions), and to treat complete security protocols (such as key-distribution protocols) in addition to basic algorithms.

---

[1] In this context, "formal" means "symbolic", as in "formal languages" and "formal logic". It does not mean "rigorous"—and indeed both views of cryptography have led to rigorous definitions and proofs.

Connections between the formal view and the computational view should ultimately benefit both:

- These connections should strengthen the foundations of formal cryptology, and help in elucidating implicit assumptions and gaps in formal methods. They should confirm or improve the relevance of formal proofs about a protocol to concrete instantiations of the protocol, making explicit requirements on the implementations of cryptographic operations.

- Methods for high-level reasoning seem necessary for computational cryptology as it treats increasingly complex systems. Formal approaches suggest such high-level reasoning principles, and even permit automated proofs. In addition, some formal approaches capture naive but powerful intuitions about cryptography; a link with those intuitions should increase the appeal and accessibility of computational cryptology.

The next section is a more detailed discussion of the two views of cryptography; it also mentions related work. The rest of the paper proceeds as follows.

In Section 3, we define a class of expressions and an equivalence relation on those expressions. The expressions represent data, of the sort used in messages in security protocols; the equivalence relation captures when two pieces of data "look the same" to an adversary, treating encryption as a formal operator. These definitions are simple and purely syntactic. In particular, they do not require any notion of probability or computational complexity. They are typical of the definitions given in formal treatments of cryptography, and directly inspired by some of them.

Then, in Section 4, we present a computational model with strings of bits, probabilities, and complexities. In this model, we define secure encryption in terms of computational indistinguishability; our definition is similar, but not identical, to those of semantic security [7, 24].

Finally, in Section 5, we relate equivalence to computational indistinguishability. We associate a probability ensemble with each formal expression; our main theorem establishes that equivalent expressions induce computationally indistinguishable ensembles. For example, the two expressions that represent two pieces of data encrypted under a fresh key will be equivalent. This equivalence can be read as a secrecy property, namely that the ciphertexts do not reveal the data. Our main theorem implies that the two expressions correspond to computationally indistinguishable ensembles.

2 Background and Related Work

This section explains the two views of cryptography, still informally. It points to a few examples of work informed by those two views; there are many more. It also describes some related research.

The formal view There is a large body of literature that treats cryptographic operations as purely formal. There, for example, the expression $\{M\}_K$ may represent an encrypted message, with plaintext M and key K. All of $\{M\}_K$, M, and K are formal expressions, rather than sequences of bits. Various functions can be applied to such expressions, yielding other expressions. One of them is decryption, which produces M from $\{M\}_K$ and K. Crucially, there is no way to recover M or K from $\{M\}_K$ alone. Thus, the idealized security properties of encryption are modeled (rather than defined). They are built into the model of computation on expressions.

This body of literature starts with the work of Dolev and Yao [17], DeMillo, Lynch, and Merritt [15], Millen, Clark, and Freedman [35], Kemmerer [30], Burrows, Abadi, and Needham [13], and Meadows [34]. It includes many different agendas and approaches, with a variety of techniques from the fields of rewriting, modal logic, process algebra, and others. Over the years, it has been used in the design of protocols, it has helped develop confidence in some existing protocols, and it has enabled the discovery of many attacks. It has also led to the development of effective methods and tools for automated protocol analysis; Lowe's and Paulson's works are two recent examples of these advances [32, 37].

This formal perspective is fairly easy to apply for the users of encryption, for example for protocol designers. It captures an important intuition: an encrypted message reveals its plaintext only to those that know the corresponding decryption key, and it reveals nothing to others. This assertion is a simple (and simplistic) all-or-nothing statement, which can be conveniently built into a formal method. In particular, it does not require any notion of probability or of computational complexity: there is no need to say that an adversary may obtain some data but only with low probability or after an expensive computation. (However, probability and computational complexity are compatible with formalism, as demonstrated by the work of Lincoln et al. [31].)

Those who employ the formal definitions often warn that a formal proof does not imply a guarantee of security. One of the reasons for this caveat is the gap between the representation of encryption in a formal model and its concrete implementation. At the very least, it is desirable to know what assumptions about encryption are necessary. Those assumptions have seldom been stated explicitly, and not in enough detail to permit systematic discussion and rigorous proofs. We aim to remedy this situation.

A somewhat similar situation arises from the use of the random-oracle model in cryptography [10]: proofs that assume random oracles do not automatically yield guarantees when the oracles are instantiated. However, we do not know of any natural examples where this gap has manifested itself (cf. [14]).

The computational view Another school of cryptographic research is based on the framework of computational complexity theory. A typical member of that school would probably say that the formal perspective is naive and disconnected from the realities of concrete cryptographic algorithms and protocols. Keys, plaintexts, and ciphertexts are all just strings of bits. An encryption function is just an algorithm. An adversary is essentially a Turing machine. Good protocols are those in which adversaries cannot do "something bad" too often and efficiently enough. These definitions are all about success probabilities and computational cost.

This computational view originates in the work of Blum and Micali [11], Yao [44], and Goldwasser and Micali [24]. It has strengthened the scientific foundations of cryptography, with a sophisticated body of definitions and theorems. It has also played a significant role in the development and study of particular protocols.

As an important example of the computational approach, we sketch a notion of secure encryption. Specifically, we choose to treat symmetric encryption, following Bellare, Desai, Jokipii, and Rogaway [7]. An encryption scheme is defined as a triple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Algorithm $\mathcal{K}$ (the key generator) makes random choices and then outputs a string k. Algorithm $\mathcal{E}$ (the encryption algorithm) flips random coins r to map strings k and m into a string $\mathcal{E}_k(m, r)$. Algorithm $\mathcal{D}$ (the decryption algorithm) maps strings k and c into a string $\mathcal{D}_k(c)$. We expect that $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$ for appropriate k, m, and r.

An adversary for an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a Turing machine which has access to an oracle. We imagine realizing this oracle in one of two ways. In the first, the oracle chooses (once and for all) a random key k, and then encrypts each query x using $\mathcal{E}_k$ and fresh random coins. In the second, the oracle chooses (once and for all) a key k, and then, when presented with a query x, encrypts a string of 0 bits of equal length, using fresh random coins. An adversary's *advantage* is the probability that the adversary outputs 1 when the oracle is realized in the first way minus the probability that the adversary outputs 1 when the oracle is realized in the second way. An encryption scheme is regarded as good if an adversary's maximal advantage is a slow-growing function of the adversary's computational resources. This definition of security can be worked out rigorously and elegantly in both asymptotic and concrete versions (see Section 4.3). In any case, it is based on notions of probability and computational power.

Related work The desire to relate the two views of cryptography is not entirely new (e.g., [3, 27, 33]). Nevertheless, there have been hardly any research efforts in this general direction. The work of Pfitzmann, Schunter, and Waidner [38] (which is simultaneous to ours and independent) starts from motivations similar to our own. It proves that some reactive, cryptographic systems satisfy high-level (non-cryptographic) specifications, under computational assumptions on cryptographic operations. These results do not concern a formal model of cryptography, such as the one studied in this paper, but the relation to a formal model of cryptography is mentioned as an interesting subject for further work. Also relevant is the work of Lincoln, Mitchell, Mitchell, and Scedrov [31], which develops a rich process-algebraic framework that draws on both views of cryptography. Further afield, Abadi, Fournet, and Gonthier [1, 2] and Lynch [33] relate the formal view of cryptography with higher-level (non-cryptographic) descriptions of security mechanisms. Finally, Volpano and Smith [42] analyze the complexity of attacking programs written in a simple, typed language; however, this language does not include cryptographic primitives.

As we compare two accounts of encryption, we arrive at the concept of which-key concealing encryption, with which ciphertexts do not manifest whether they were produced using the same key (see Section 4.2). Independently and concurrently, the work of M. Fischlin and that of Bellare, Boldyreva, Desai, and Pointcheval study this concept from different perspectives [6, 19].

3 Formal Encryption and Expression Equivalence

In this section we present the formal view of cryptography, specifically treating symmetric encryption. We describe the space of expressions on which encryption operates, and what it means for two expressions to be equivalent.

As explained in the introduction, the expressions represent data, of the sort used in messages in security protocols. Expressions are built up from bits and keys by pairing and encryption. The equivalence relation captures when two pieces of data "look the same" to an adversary that has no prior knowledge of the keys used in the data. For example, an adversary (with no prior knowledge) cannot obtain the key K from the ciphertexts $\{0\}_K$ and $\{1\}_K$; therefore, the adversary cannot decrypt and distinguish these ciphertexts, so they are equivalent. Similarly, the pairs $(0, \{0\}_K)$ and $(0, \{1\}_K)$ are equivalent. On the other hand, the pairs $(K, \{0\}_K)$ and $(K, \{1\}_K)$ are not equivalent, since an adversary can obtain K from them, then decrypt $\{0\}_K$ or $\{1\}_K$ and obtain 0 or 1, respectively, thus distinguishing the pairs. In this section, we formalize these informal arguments about equivalence; the soundness theorem of Section 5 provides a further justification for them.

3.1 Expressions

We write **Bool** for the set of bits $\{0, 1\}$. These bits can be used to spell out numbers and principal names, for example. We write **Keys** for a fixed, nonempty set of symbols disjoint from **Bool**. The symbols $K, K', K'', \ldots$ and $K_1, K_2, \ldots$ are all in **Keys**. Informally, elements of the set **Keys** represent cryptographic keys, generated randomly by a principal that is constructing an expression. Formally, however, keys are atomic symbols, not strings of bits. We write **Exp** for the set of *expressions* defined by the grammar:[2]

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

Informally, (M, N) represents the pairing of M and N, which might be implemented by concatenation plus markers, and $\{M\}_K$ represents the encryption of M under K, which might be implemented using a symmetric algorithm like DES, in CBC mode and with a random initialization vector. Pairing and encryption can be nested, as in the expression $(\{\{(0, K')\}_K\}_{K'}, K)$.

[2] An equivalent way to define **Exp** is as the language generated by the context-free grammar with start symbol **E**, nonterminals **E** and **K**, terminals "0", "1", "(", ")", ",", "{", "}", and the set of elements in **Keys**, and the productions:

$$\mathbf{E} \longrightarrow 0 \mid 1 \mid (\mathbf{E}, \mathbf{E}) \mid \mathbf{K} \mid \{\mathbf{E}\}_{\mathbf{K}}$$
$$\mathbf{K} \longrightarrow K \quad \text{for each } K \in \mathbf{Keys}$$

We emphasize that the elements of **Exp** are formal expressions (essentially, parse trees, abstract syntax trees) rather than actual keys, bits, concatenations, or encryptions. In particular, they are unambiguous: for example, (M, N) equals (M', N') if and only if M equals M' and N equals N', and it never equals $\{M'\}_K$. Similarly, $\{M\}_K$ equals $\{M'\}_{K'}$ if and only if M equals M' and K equals K'. However, according to definitions given below, $\{M\}_K$ and $\{M'\}_{K'}$ are equivalent even when M and M' are different and when K and K' are different.

There are several possible extensions of the set of expressions:

- We could allow expressions of the form $\{M\}_N$, where an arbitrary expression N is used as encryption key.
- We could distinguish encryption keys from decryption keys, as in public-key cryptosystems.

These extensions are useful in modeling realistic protocols, but would complicate our definitions and theorems. We therefore leave them for further work.

It is also important to consider a restriction to the set of expressions. 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. For example, K encrypts both K_1 and K_2 in $(\{\{K_1\}_{K_2}\}_K, 0)$. For each M, this defines a binary relation on keys, the "encrypts" relation. (As a variant, a more liberal definition that ignores occurrences of K' as a subscript may also be adequate for our purposes.) We say that M is *cyclic* (or *acyclic*) if its associated "encrypts" relation is cyclic (or acyclic, respectively). For example, $\{K\}_K$ and $(\{K\}_{K'}, \{K'\}_K)$ are both cyclic, while $(\{K\}_{K'}, \{0\}_K)$ is acyclic.

Cycles, such as encrypting a key under itself, are a source of errors in practice (e.g., [43]); they also lead to weaknesses in common computational models, as explained in Section 4. Moreover, cycles can often be avoided in practice—and they should generally be avoided given what is, and is not, known about them. The soundness theorem of Section 5 deals only with acyclic expressions. In contrast, cycles are typically permitted (without discussion!) in formal methods.

3.2 Equivalence

Next we give a formal definition of equivalence of expressions. It draws on definitions from the works of Syverson and van Oorschot, Schneider, Paulson, and others [37, 39, 40]. Some of the auxiliary definitions concern how expressions can be analyzed and synthesized; such definitions are quite common in formal methods for protocol analysis. Equivalence relations are useful in semantics of modal logics: in such semantics, one says that two states in a computation "look the same" to a principal only if the principal has equivalent expressions in those states. Equivalence relations also appear in bisimulation proof techniques [4, 12], where one requires that bisimilar processes produce equivalent messages.

First, we define an *entailment* relation $M \vdash N$, where M and N are expressions. Intuitively, $M \vdash N$ means that N can be computed from M. Formally, we define the relation inductively, as the least relation with the following properties:

- $M \vdash 0$ and $M \vdash 1$,
- $M \vdash M$,
- if $M \vdash N_1$ and $M \vdash N_2$ then $M \vdash (N_1, N_2)$,
- if $M \vdash (N_1, N_2)$ then $M \vdash N_1$ and $M \vdash N_2$,
- if $M \vdash N$ and $M \vdash K$ then $M \vdash \{N\}_K$,
- if $M \vdash \{N\}_K$ and $M \vdash K$ then $M \vdash N$.

This definition of $M \vdash N$ models what an attacker can obtain from M without any prior knowledge of the keys used in M. For example, we have

$$(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \vdash K_3$$

and

$$(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \vdash \{K_1\}_{K_2}$$

but not

$$(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \vdash K_1 \qquad \text{(false)}$$

It is simple to derive a more general definition from this one: obtaining N from M with prior knowledge of K is equivalent to obtaining N from (M, K) with no prior knowledge.

Next, we introduce the box symbol $\square$, which represents a ciphertext that an attacker cannot decrypt. We define the set **Pat** of *patterns* as an extension of the set of expressions, with the grammar:[3]

$P, Q ::=$		patterns
	K	key (for $K \in \mathbf{Keys}$)
	i	bit (for $i \in \mathbf{Bool}$)
	(P, Q)	pair
	$\{P\}_K$	encryption (for $K \in \mathbf{Keys}$)
	$\Box$	undecryptable

Intuitively, a pattern is an expression that may have some parts that an attacker cannot decrypt.

We define a function that, given a set of keys T and an expression M, reduces M to a pattern. Intuitively, this is the pattern that an attacker can see in M if the attacker has the keys in T.

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

Further, we define a pattern for an expression without an auxiliary set T, but using the set of keys obtained from the expression itself.

$$\mathit{pattern}(M) \ = \ p(M, \{K \in \mathbf{Keys} \mid M \vdash K\})$$

Intuitively, this is the pattern that an attacker can see in M using the set of keys obtained from M. (As above, we assume that the attacker has no prior knowledge of the keys used in M, without loss of generality.) For example, we have

$$\mathit{pattern}((\{\{K_1\}_{K_2}\}_{K_3}, K_3)) \ = \ (\{\Box\}_{K_3}, K_3)$$

because $\{K \in \mathbf{Keys} \mid (\{\{K_1\}_{K_2}\}_{K_3}, K_3) \vdash K\}$ is $\{K_3\}$ and

$$p((\{\{K_1\}_{K_2}\}_{K_3}, K_3), \{K_3\}) \ = \ (\{\Box\}_{K_3}, K_3)$$

[3] An equivalent way to define **Pat** is as the language generated by the context-free grammar with start symbol **P**, nonterminals **P** and **K**, terminals "0", "1", "(", ")", ",", "{", "}", "□ ", and the set of elements in **Keys**, and the productions:

$$\begin{array}{l} \mathbf{P} \longrightarrow 0 \mid 1 \mid (\mathbf{P}, \mathbf{P}) \mid \mathbf{K} \mid \{\mathbf{P}\}_{\mathbf{K}} \mid \Box \\ \mathbf{K} \longrightarrow K \quad \text{for each } K \in \mathbf{Keys} \end{array}$$

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

$$M \equiv N \;\text{ if and only if }\; \mathit{pattern}(M) = \mathit{pattern}(N)$$

For example, we have:

$$(\{\{K_1\}_{K_2}\}_{K_3}, K_3) \equiv (\{\{0\}_{K_1}\}_{K_3}, K_3)$$

since both expressions yield the pattern $(\{\Box\}_{K_3}, K_3)$.

We may view keys as bound names, subject to renaming (as in the spi calculus [5]). For example, although $(\{0\}_K, K)$ and $(\{0\}_{K'}, K')$ are not equivalent, we may say that they are equivalent up to renaming. More generally, we define *equivalence up to renaming*, $\cong$, as follows:

$$M \cong N \quad \text{if and only if} \quad \begin{array}{l}\text{there exists a bijection } \sigma \text{ on } \mathbf{Keys}\\ \text{such that } M \equiv N\sigma\end{array}$$

where $N\sigma$ is the result of applying σ as a substitution to N. Although this relation $\cong$ is looser than $\equiv$, our soundness theorem treats it smoothly, without difficulty. Therefore, we focus on $\cong$. In informal discussions, we often do not distinguish the two relations, calling them both equivalence.

3.3 Some examples and some subtleties

In this section we give a few more examples. Some of the examples indicate assumptions and choices built into the definition of equivalence. These are fairly subtle but important, and it is useful to be explicit about them. We revisit them in Section 4.

- $0 \cong 0$, of course.
- $0 \not\cong 1$, of course.
- $\{0\}_K \cong \{1\}_K$.
- $(K, \{0\}_K) \not\cong (K, \{1\}_K)$, but $(K, \{(\{0\}_{K'}, 0)\}_K) \cong (K, \{(\{1\}_{K'}, 0)\}_K)$.
- $K \not\equiv K'$ and $K \cong K'$, since keys are subject to renaming with $\cong$ but not with $\equiv$.
- $\{0\}_K \cong \{1\}_{K'}$ and even $\{0\}_K \equiv \{1\}_{K'}$, although the two ciphertexts are under different keys.
- $(\{K'\}_K, \{0\}_K) \cong (\{K'\}_K, \{1\}_{K'})$ and even $(\{K'\}_K, \{0\}_K) \equiv (\{K'\}_K, \{1\}_{K'})$, similarly.

- $\{0\}_K \cong \{K\}_K$, despite the encryption cycle in $\{K\}_K$.

- $\{(((1,1),(1,1)),((1,1),(1,1)))\}_K \cong \{0\}_K$.

  Informally, we are assuming that a plaintext of any size can be encrypted, and that the size of the plaintext cannot be deduced from the resulting ciphertext without knowledge of the corresponding decryption key. This property justifies equivalences such as the one above, where the two plaintexts are of different sizes. In an implementation, it can be guaranteed by padding plaintexts up to a maximum size, and truncating larger expressions or mapping them to some fixed string (see Section 4).

  We could easily refine the equivalence relation to make it sensitive to sizes, for example by introducing a symbol $\Box_n$ for each size n. The resulting definitions would be heavier.

- $(\{0\}_K, \{0\}_K) \cong (\{0\}_K, \{1\}_K)$.

  Informally, we are assuming that an attacker who does not have a key cannot even detect whether two plaintexts encrypted under the key are identical. For example, the attacker should not be able to tell that the same plaintext appears twice under K in $(\{0\}_K, \{0\}_K)$, hence $(\{0\}_K, \{0\}_K) \cong (\{0\}_K, \{1\}_K)$. In an implementation, this sort of equivalence can be guaranteed by randomization of the encryption function (see Section 4).

  We could easily refine the equivalence relation to make it sensitive to message identities (for example as in [4]); but, again, the resulting definitions would be heavier.

- $(\{0\}_K, \{1\}_K) \cong (\{0\}_K, \{1\}_{K'})$.

  Informally, we are assuming that an attacker who does not have a key cannot even detect whether two ciphertexts use that same key. For example, the attacker should not be able to tell that the same key is used twice in $(\{0\}_K, \{1\}_K)$, hence $(\{0\}_K, \{1\}_K) \cong (\{0\}_K, \{1\}_{K'})$.

  Again, an alternative definition would be possible, with some complications.

4 The Computational View: Encryption Schemes and Indistinguishability

In this section we provide a computational treatment for symmetric encryption (adapting those of [7, 24]). First we describe the functions that constitute a symmetric encryption scheme, and then we describe when an encryption scheme should be called secure. Actually, there are a few different possibilities for defining security, and we discuss several of them. The notion that we focus on—which we call type-0 security—is stronger than the customary notion of security (that is, semantic security, and notions equivalent to it [7, 20, 24]). Nonetheless, one can achieve type-0 security under standard complexity-theoretic assumptions. We focus on type-0 security because it matches up with the formal definitions of Section 3. Other computational notions of security can be paired with analogous formal ones.

4.1 Preliminaries

Elements of an encryption scheme Let $\mathsf{String} = \{0,1\}^*$ be the set of all finite strings, and let $|x|$ be the length of string x. Let $\mathsf{Plaintext}$, $\mathsf{Ciphertext}$, and Key be nonempty sets of finite strings. Let $\mathbf{0}$ be a particular string in $\mathsf{Plaintext}$. Encrypting a string not in $\mathsf{Plaintext}$ will result in a ciphertext that decrypts to $\mathbf{0}$. We assume that if $x \in \mathsf{Plaintext}$ then $x' \in \mathsf{Plaintext}$ for all x' of the same length as x. Let Key be endowed with some fixed distribution. (If Key is finite, the distribution on Key is the uniform one.) Let Coins be a synonym for $\{0,1\}^\omega$ (the set of infinite strings), and $\mathsf{Parameter}$ (the set of *security parameters*) be a synonym for 1^* (the set of finite strings of 1 bits).

An *encryption scheme*, Π, is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, where

$$\begin{aligned}
\mathcal{K} &: \mathsf{Parameter} \times \mathsf{Coins} \to \mathsf{Key} \\
\mathcal{E} &: \mathsf{Key} \times \mathsf{String} \times \mathsf{Coins} \to \mathsf{Ciphertext} \\
\mathcal{D} &: \mathsf{Key} \times \mathsf{String} \to \mathsf{Plaintext}
\end{aligned}$$

and each algorithm is computable in time polynomial in the size of its input (but without consideration for the size of Coins input). Algorithm $\mathcal{K}$ is called the *key-generation* algorithm, $\mathcal{E}$ is called the *encryption* algorithm, and $\mathcal{D}$ is called the *decryption* algorithm. We usually write the first argument to $\mathcal{E}$ or $\mathcal{D}$, the key, as a subscript. When we omit mention of the final argument to $\mathcal{K}$ or $\mathcal{E}$ this indicates the corresponding probability space, or, when used as a set, the support of that probability space (that is, the strings which are output with nonzero probability). We require that for all $\eta \in \mathsf{Parameter}$,

$k \in \mathcal{K}(\eta)$, and $r \in \mathsf{Coins}$, if $m \in \mathsf{Plaintext}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$, while if $m \notin \mathsf{Plaintext}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = \mathbf{0}$. For example, the encryption function could treat an out-of-domain message as though it was $\mathbf{0}$. We insist that $|\mathcal{E}_k(x)|$ depends only on η and $|x|$ when $k \in \mathcal{K}(\eta)$.

The definition above is for probabilistic, stateless encryption. One can be a bit more general, allowing the encryption algorithm to maintain state. We do not pursue this generalization here.

Other basic concepts A function $\epsilon : \mathsf{N} \to \mathsf{R}$ is *negligible* if $\epsilon(\eta) \in \eta^{-\omega(1)}$. This means that for all $c > 0$ there exists N_c such that $\epsilon(\eta) \leq \eta^{-c}$ for all $\eta \geq N_c$. An *ensemble* (or *probability ensemble*) is a collection of distributions on strings, $D = \{D_\eta\}$, one for each η. We write $x \stackrel{R}{\leftarrow} D_\eta$ to indicate that x is sampled from D_η, and write $\Pr[x \stackrel{R}{\leftarrow} D_\eta : E]$ for the probability of event E when x is sampled from D_η. Let $D = \{D_\eta\}$ and $D' = \{D'_\eta\}$ be ensembles. We say that D and D' are *indistinguishable* (or *computationally indistinguishable*), and write $D \approx D'$, if for every probabilistic polynomial-time adversary A, the function

$$\epsilon(\eta) \stackrel{\text{def}}{=} \Pr[x \stackrel{R}{\leftarrow} D_\eta :\ A(\eta,\ x) = 1] - \Pr[x \stackrel{R}{\leftarrow} D'_\eta :\ A(\eta,\ x) = 1]$$

is negligible.

4.2 Aspects of encryption-scheme security

In this section we consider some possible attributes of encryption schemes, and also consider encryption cycles. These issues already appear in Section 3 in a formal setting; here we explore them further in a computational setting.

Attributes (present or absent) of a secure encryption scheme We single out three characteristics of an encryption scheme. The first and third are well-known, while the second seems not to have received attention till now.

- Repetition concealing vs. repetition revealing

  Given ciphertexts c and c', can one tell if their underlying plaintexts are equal? If so, we call the scheme repetition revealing; otherwise, it is repetition concealing. A repetition-concealing scheme must be probabilistic (or stateful); making encryption schemes repetition concealing is one motivation for probabilistic encryption [24].

- Which-key concealing vs. which-key revealing

  If one encrypts messages under various keys, can one tell which messages were encrypted under the same keys? If so, we call the scheme which-key revealing; otherwise, it is which-key concealing. Though some standard instantiations of encryption schemes are which-key concealing, others are not (for example, standard asymmetric encryption methods based on RSA), and standard definitions for encryption-scheme security (like those in [7, 24]) do not guarantee which-key concealment. Demanding that an encryption scheme be which-key concealing is useful in contexts beyond that of the present paper (for example, in achieving forms of anonymity). The current work of Bellare et al. undertakes a thorough treatment of which-key concealing encryption [6].

- Message-length concealing vs. message-length revealing

  Does a ciphertext reveal the length of its underlying plaintext? If so, we call the scheme message-length revealing; otherwise, it is message-length concealing. Most encryption schemes are message-length revealing. The reason is that implementing message-length concealing encryption invariably entails padding messages to some maximal length, and it may therefore be quite inefficient. Message-length concealing encryption is possible when the message space is finite, or when all ciphertexts are infinite streams (rather than finite strings as stated in our definitions).

These three characteristics are orthogonal, and all eight combinations make sense. Let us call these eight notions of security type-0, type-1, ..., type-7, with the numbering determined as follows: concealing corresponds to a 0 bit and revealing to a 1 bit, and we interpret the three characteristics above as a 3-bit binary number, the most significant bit being for repetition concealing or revealing, then which-key concealing or revealing, finally message-length concealing or revealing. With this terminology, the conventional concept of encryption-scheme security, ever since the work of Goldwasser and Micali [24], has been type-3 security: a ciphertext may reveal the length of the message and which key is being used, but it should not reveal if two ciphertexts are encryptions of the same message. However, this concept of security is not the only reasonable one.

Encryption cycles Given a type-n ($n \in \{0, \ldots, 7\}$) secure encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, one can construct a type-n secure encryption scheme

$\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ with the following property: Π' would be completely insecure if the adversary were given (for example, as an additional input) even a single encryption $c \stackrel{R}{\leftarrow} \mathcal{E}'_k(k)$ of the underlying key k. Goldwasser and Micali were aware of this (in the public-key setting) when they published their work [24].

It is not only encrypting k under k that is problematic; longer cycles may also cause problems. For example, even if an encryption scheme is type-3 secure, it may not be safe to encrypt a message b under a key a and then, reversing the roles of a and b, to encrypt a under b. For all we know, the concatenation of the two ciphertexts might trivially reveal both a and b. For probabilistic encryption, for cycles of length greater than one, we do not have any example to demonstrate that this problem can actually arise, but the hybrid arguments [24] often used to prove encryption schemes secure, and which we use here, do not work in the presence of such cycles.

Therefore, as discussed in Section 3, we focus on expressions without encryption cycles. In return, we can rely on standard-looking definitions and tools in the computational setting.

4.3 Definitions of encryption-scheme security (types 0, 1, 3)

The formal treatment in Section 3 corresponds to type-0 security (repetition concealing, which-key concealing, and message-length concealing), so let us define this notion more precisely. An explanation of the notation follows the definition.

Definition 1 (Type-0 security) *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let $\eta \in$ Parameter be a security parameter, and let A be an adversary. Define*

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

Encryption scheme Π is type-0 secure *if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}^0_{\Pi[\eta]}(A)$ is negligible (as a function of η).*

We are looking at the difference of two probabilities.

- First, let us focus on the first probability. The quantity in brackets describes an experiment that is performed, and then an event. In this experiment, one first chooses two keys, k and k', independently, by running the key-generation algorithm $\mathcal{K}$. Then one runs adversary A, with two oracles: a left oracle f and a right oracle g. If the adversary asks the left oracle f a query $m \in \mathsf{String}$, the oracle returns a random encryption of m under key k. That is, the oracle computes $c \xleftarrow{R} \mathcal{E}_k(m)$ and returns c. If the adversary asks the right oracle g a query $m \in \mathsf{String}$, the oracle returns a random encryption of m under key k', similarly. Independent coins are used each time a string is encrypted (but the keys k and k' stay fixed).

- Next, let us consider the second probability. In this experiment, a single key k is selected by running the key-generation algorithm $\mathcal{K}$. The adversary again has two oracles, a left oracle f and a right oracle g, and these oracles again expect queries $m \in \mathsf{String}$. But now the oracles behave in the same way. When asked a query m, the oracles ignore the query, sample $c \xleftarrow{R} \mathcal{E}_k(\mathbf{0})$, and return c. Independent coins are used each time a string is encrypted (but the key k stays fixed).

The *type-0 advantage* is the difference in the above probabilities. One can imagine that the adversary is trying to distinguish a good encryption box from a false one. A good encryption box encrypts the specified query using the selected key. A false encryption box ignores the query and encrypts a fixed message under a fixed random key. Intuitively, a scheme is type-0 secure if no computationally efficient adversary can do a good job of telling apart the two encryption boxes on the basis of their input/output behavior.

Various other equivalent formalizations for type-0 encryption are possible. For example, it adds no power for there to be more than two oracles. (In the first experiment, each oracle would encrypt queries under its own key; in the second, every oracle would encrypt $\mathbf{0}$ under a common key.) Likewise, it takes away no power if $\mathcal{E}_{k'}(\cdot)$ is replaced with $\mathcal{E}_{k'}(\mathbf{0})$ in the first experiment.

We also give detailed definitions of type-1 and type-3 security. (Recall that type-1 means repetition concealing and which-key concealing, and that type-3 means repetition concealing.) The definitions resemble that of type-0 security. In these definitions, $\mathcal{E}_k(\cdot)$ is an oracle that returns $c \xleftarrow{R} \mathcal{E}_k(m)$ on input m, as above, and $\mathcal{E}_k(0^{|\cdot|})$ is an oracle that returns $c \xleftarrow{R} \mathcal{E}_k(0^{|m|})$ on input m.

Definition 2 (Type-1 security) *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let $\eta \in$ $\mathsf{Parameter}$ be a security parameter, and let A be an adversary. Define*

$$\mathsf{Adv}^{1}_{\Pi[\eta]}(A) \quad \stackrel{\text{def}}{=} \quad \Pr\left[k, k' \xleftarrow{R} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)}(\eta) = 1\right] -$$

$$\Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(0^{|\cdot|}),\mathcal{E}_k(0^{|\cdot|})}(\eta) = 1\right]$$

Encryption scheme Π *is* type-1 secure *if for every probabilistic polynomial-time adversary* A, $\mathsf{Adv}^1_{\Pi[\eta]}(A)$ *is negligible (as a function of* η*).*

Definition 3 (Type-3 security) *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an encryption scheme, let* $\eta \in \mathsf{Parameter}$ *be a security parameter, and let* A *be an adversary. Define*

$$\begin{aligned}\mathsf{Adv}^3_{\Pi[\eta]}(A) \quad \stackrel{\text{def}}{=} \quad & \Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(\cdot)}(\eta) = 1\right] - \\ & \Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(0^{|\cdot|})}(\eta) = 1\right]\end{aligned}$$

Encryption scheme Π *is* type-3 secure *if for every probabilistic polynomial-time adversary* A, $\mathsf{Adv}^3_{\Pi[\eta]}(A)$ *is negligible (as a function of* η*).*

4.4 Achieving type-0 and type-1 security with standard tools

Since type-3 security is standard but type-0 and type-1 security are not, we show that type-0 and type-1 security can be achieved using standard assumptions and constructions. Although this fact is not necessary for our soundness theorem, it provides support for the hypotheses of the theorem.

Block ciphers Let $\beta \geq 1$ be a number (the blocksize) and let $\mathsf{Block} = \{0,1\}^\beta$. Let Key be a finite nonempty set. Then a *block cipher* is a function $E : \mathsf{Key} \times \mathsf{Block} \to \mathsf{Block}$ such that, for every $k \in \mathsf{Key}$, we have that $E_k(\cdot) = E(k, \cdot)$ is a permutation. Example block ciphers are DES and the emerging AES (Advanced Encryption Standard).

One measure of security for a block cipher is:

$$\mathsf{Adv}^{\mathrm{prp}}_E(A) \quad = \quad \Pr\left[k \xleftarrow{R} \mathsf{Key} :\ A^{E_k(\cdot)} = 1\right] - \Pr\left[\pi \xleftarrow{R} \mathrm{Perm}(\beta) :\ A^{\pi(\cdot)} = 1\right]$$

Here $\mathrm{Perm}(\beta)$ denotes the set of all permutations on $\{0,1\}^\beta$. Informally, the adversary A is trying to distinguish the block cipher E, as it behaves on a random key k, from a random permutation π. We think of E as a good block cipher if $\mathsf{Adv}^{\mathrm{prp}}_E(A)$ is small as long as A is of reasonable computational complexity.

Block cipher modes of operation Block ciphers are the most common building block for making symmetric encryption schemes. Two well-known ways to do this are CBC mode and CTR mode. In CBC mode (with a random initialization vector), the encryption of a plaintext $x = x_1 \ldots x_n$ using key $k \in \mathsf{Key}$, where $n \geq 1$ and $|x_i| = \{0,1\}^\beta$, is $y_0 y_1 \ldots y_n$ where $y_0 \stackrel{R}{\leftarrow} \mathsf{Block}$ and $y_i = E_k(y_{i-1} \oplus x_i)$ for all $1 \leq i \leq n$. In CTR mode, the encryption of a plaintext x using key k is the concatenation of $r \stackrel{R}{\leftarrow} \mathsf{Block}$ with the xor of x and the $|x|$-bit prefix of the concatenation of $E_k(r)$, $E_k(r+1)$, $E_k(r+2), \ldots$. Here $r+i$ is the β-bit string that encodes the sum of r (treated as an unsigned number) and i, modulo 2^β. In [7], Bellare et al. establish the (type-3) security of these two modes of operation.

CBC and CTR modes are which-key concealing Even though the results just mentioned do not indicate that CBC mode or CTR mode are which-key concealing, these schemes are in fact which-key concealing and those results can be used to show it, as we now sketch. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let A be an adversary, and define

$$\begin{aligned} \mathsf{Adv}^{\mathrm{rand}}_{\Pi[\eta]} &= \Pr\left[k \stackrel{R}{\leftarrow} \mathsf{Key}(\eta) :\ A^{\mathcal{E}_k(\cdot)}(\eta) = 1\right] - \\ &\quad \Pr\left[k \stackrel{R}{\leftarrow} \mathsf{Key}(\eta) :\ A^{\$^{|\mathcal{E}_k(\cdot)|}}(\eta) = 1\right] \end{aligned}$$

By $\$^{|\mathcal{E}_k(\cdot)|}$ we denote an oracle which, on input m, computes $c \stackrel{R}{\leftarrow} \mathcal{E}_k(m)$ and returns a random string of length $|c|$. (By an assumption stated above, $|c|$ depends only on η and $|m|$.) Informally, the adversary cannot tell if it is given a real encryption oracle or an oracle that returns a random string (of the appropriate length) in response to every query.

The proofs of security in [7] actually establish that CBC mode and CTR mode are good schemes according to $\mathsf{Adv}^{\mathrm{rand}}$, assuming that the underlying block cipher E is secure according to $\mathsf{Adv}^{\mathrm{prp}}$. To complete the picture, we claim that any good scheme according to $\mathsf{Adv}^{\mathrm{rand}}$ is also type-1 secure. (This claim is not hard to prove, though we omit doing so here.) Therefore, CBC mode and CTR mode (as defined above) are type-1 secure: repetition concealing, which-key concealing, but message-length revealing.

Hiding message lengths for type-0 security Finally, we have to conceal message lengths. This step is standard, provided the message space is finite. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a type-1 secure encryption scheme with $\mathsf{Plaintext} = \{0,1\}^*$. Let $\mathsf{Plaintext}' \subseteq \mathsf{String}$ be a finite set, with a particular element $\mathbf{0}'$. To make a type-0 secure encryption scheme we just encode all messages of $\mathsf{Plaintext}'$ into strings of some fixed length, and then encrypt these using $\mathcal{E}$. That is, we choose any convenient function *encode*$(\cdot)$ which (reversibly) takes strings in $\mathsf{Plaintext}'$ to a subset of $\{0,1\}^\ell$, for some number ℓ. The encryption scheme $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$, with message space $\mathsf{Plaintext}'$, is defined by letting $\mathcal{E}'_k(m) = \mathcal{E}_k(encode(m))$ for $m \in \mathsf{Plaintext}'$, setting $\mathcal{E}'_k(m) = \mathcal{E}'_k(\mathbf{0}')$ for $m \notin \mathsf{Plaintext}'$, and defining $\mathcal{D}'$ in the obvious way. Type-1 security of Π immediately implies type-0 security of Π'.

5 The Computational Soundness of Formal Equivalence

In this section we relate the two views of cryptography. We proceed in two steps. First, we show how to associate an ensemble to an expression M, given an encryption scheme Π. Then we show that, under appropriate assumptions, equivalent expressions give rise to indistinguishable ensembles.

5.1 Associating an ensemble to an expression

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $\eta \in \mathsf{Parameter}$ be a security parameter. We associate to each formal expression $M \in \mathbf{Exp}$ a distribution on strings $[\![M]\!]_{\Pi[\eta]}$, and thereby an ensemble $[\![M]\!]_{\Pi}$. This association constitutes a concrete semantics for expressions (in the style of programming-language semantics or logic semantics); it works as follows:

- First, we map each key symbol K that occurs in M to a string of bits $\tau(K)$, using the key generator $\mathcal{K}(\eta)$.
- We map the formal bits 0 and 1 to standard string representations for them.
- We obtain the image of a formal pair (M, N) by concatenating the images of the components M and N.
- We obtain the image of a formal encryption $\{M\}_K$ by calculating $\mathcal{E}_{\tau(K)}(x)$, where x is the image of M.
- In all cases, we tag string representations with their types (that is, with the strings "key", "bool", "pair", "ciphertext") in order to avoid any ambiguities.

This association is defined more precisely in Figure 1. In the figure, we write $Keys(M)$ for the set of all key symbols that occur in M, and write $\langle\, x_1, \ldots, x_k \,\rangle$ for an ordinary string encoding of the tuple $x_1, \ldots, x_k$. The auxiliary initialization procedure $\textsc{Initialize}_\eta(M)$ maps every key symbol in $Keys(M)$ to a unique, independently chosen key $\tau(K)$. The probability

algorithm $\text{INITIALIZE}_\eta(M)$
  for $K \in \mathit{Keys}(M)$ **do** $\tau(K) \xleftarrow{R} \mathcal{K}(\eta)$

algorithm $\text{CONVERT}(M)$
  if $M = K$ where $K \in$ **Keys then**
    return $\langle\, \tau(K)$, "key" $\rangle$
  if $M = b$ where $b \in$ **Bool then**
    return $\langle\, b$, "bool" $\rangle$
  if $M = (M_1, M_2)$ **then**
    return $\langle\, \text{CONVERT}(M_1), \text{CONVERT}(M_2)$, "pair" $\rangle$
  if $M = \{M_1\}_K$ **then**
    $x \xleftarrow{R} \text{CONVERT}(M_1)$
    $y \xleftarrow{R} \mathcal{E}_{\tau(K)}(x)$
    return $\langle\, y$, "ciphertext" $\rangle$

Figure 1: *How to map (probabilistically) an expression M to a string $\text{CONVERT}(M)$, given an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a security parameter η.*

of a string in $[\![M]\!]_{\Pi[\eta]}$ is that induced by the algorithm $\text{CONVERT}(M)$ of Figure 1.

5.2 Equivalence implies indistinguishability

Next we prove that equivalent expressions correspond to indistinguishable ensembles (that is, $M \cong N$ implies $[\![M]\!]_\Pi \approx [\![N]\!]_\Pi$), assuming that the expressions are acyclic and that the underlying encryption scheme is type-0 secure.

We start with a few simple examples, instantiating the claim that $M \cong N$ implies $[\![M]\!]_\Pi \approx [\![N]\!]_\Pi$.

- Since $0 \cong 0$, we conclude that $[\![0]\!]_\Pi \approx [\![0]\!]_\Pi$. The two ensembles being compared put all the probability mass on a single point, $\langle\, 0$, "bool" $\rangle$.
- Since $K \cong K'$, we conclude that $[\![K]\!]_\Pi \approx [\![K']\!]_\Pi$. The two ensembles being compared are identical: they are induced by the key generator $\mathcal{K}$.
- Since $\{0\}_K \cong \{1\}_K$, we conclude that $[\![\{0\}_K]\!]_\Pi \approx [\![\{1\}_K]\!]_\Pi$. This indistinguishability is nontrivial: it relies on the assumption that the encryption scheme is type-0 secure.

- Although $\{0\}_K \cong \{K\}_K$, we cannot conclude anything about how $[\![\{0\}_K]\!]_\Pi$ may relate to $[\![\{K\}_K]\!]_\Pi$, because of the encryption cycle in $\{K\}_K$.

Reconsidering some of the other examples of Section 3.3 can also be instructive.

Our theorem is:

Theorem 1 *Let M and N be acyclic expressions and let Π be a type-0 secure encryption scheme. Suppose that $M \cong N$. Then $[\![M]\!]_\Pi \approx [\![N]\!]_\Pi$.*

Proof The proof is a hybrid argument, as in [24]. (See [21] for a disucssion of this technique.) One must be particularly careful in forming the hybrids, relying on acyclicity. Because of the generality of the claim, the description of the hybrid argument is somewhat complex. Therefore, we include a running example, in *italics*. We also break up the proof into several phases.

Key renaming The first phase of the proof deals with renaming keys. Roughly, its goal is to modify the expressions M and N by renaming keys so that $\mathit{pattern}(M) = \mathit{pattern}(N)$, still, and M has "hidden" keys $K_1, \ldots, K_m$ and N has "hidden" keys $K_1, \ldots, K_n$, where K_I encrypts K_i only when $I \geq i$, and both M and N have "recoverable" keys $J_1, \ldots, J_\mu$.

As above, $\mathit{Keys}(M)$ is the set of all keys that occur in M. First we partition $\mathit{Keys}(M)$, separating those keys that the adversary can recover from the rest:

$$\begin{aligned}\mathit{recoverable}(M) &= \{K \in \mathit{Keys}(M) \mid M \vdash K\} \\ \mathit{hidden}(M) &= \mathit{Keys}(M) - \mathit{recoverable}(M)\end{aligned}$$

Let $\mu = |\mathit{recoverable}(M)|$ and let $m = |\mathit{hidden}(M)|$. We form a directed graph $G_M = (V_M, E_M)$ whose vertices are $V_M = \mathit{hidden}(M)$ and where there is an arc $K \to K'$ in E_M if and only if K encrypts K' in M. (Recall that K encrypts K' in M if there is a subexpression $\{M_1\}_K$ of M where K' occurs in M_1.) The acyclicity of M means that G_M is acyclic and, as a consequence, we can rename the keys in $\mathit{Keys}(M)$ so that the hidden keys are $K_1, \ldots, K_m$, the recoverable keys are $J_1, \ldots, J_\mu$, and $K_I \to K_i \in E_M$ implies $I \geq i$. In other words, a deeper key of M gets a smaller number. We let M' be the resulting expression.

Let us start our running example. Suppose that M is the expression:

$$\{0\}_{K_6}\ \{K_1\ 1\}_{K_4}\ K_2\ \{0\}_{K_3}\ \{K_6\}_{K_4}\ \{K_1\ K_3\}_{K_4}\ \{1\ 1\ 1\}_{K_5}\ 0\ \{K_1\}_{K_6}\ \{K_5\}_{K_2}$$

Here we have omitted commas, for readability, and also parentheses. The parentheses are irrelevant in this example, but we should hold them fixed as the example is developed; for concreteness, we may think of elements as being grouped left-to-right, so that a b c d is short for $(((a, b), c), d)$. *In this example, we have:*

$$\begin{aligned} \mathit{Keys}(M) &= \{K_1, K_2, K_3, K_4, K_5, K_6\} \\ \mathit{recoverable}(M) &= \{K_2, K_5\} \\ \mathit{hidden}(M) &= \{K_1, K_3, K_4, K_6\} \end{aligned}$$

The graph $G_M = (V_M, E_M)$ *has vertices* $V_M = \{K_1, K_3, K_4, K_6\}$ *and arcs* $K_4 \to K_1$, $K_4 \to K_3$, $K_4 \to K_6$, *and* $K_6 \to K_1$. *We rename* $(K_1, K_2, K_3, K_4, K_5, K_6)$ *to* $(K_1, J_1, K_2, K_4, J_2, K_3)$, *obtaining a new expression* M'*:*

$$\{0\}_{K_3}\ \{K_1\ 1\}_{K_4}\ J_1\ \{0\}_{K_2}\ \{K_3\}_{K_4}\ \{K_1\ K_2\}_{K_4}\ \{1\ 1\ 1\}_{J_2}\ 0\ \{K_1\}_{K_3}\ \{J_2\}_{J_1}$$

Because $M \cong N$, and by the definition of equivalence up to renaming, there exists a function σ on $\mathit{Keys}(N)$ such that $\mathit{pattern}(M) = \mathit{pattern}(N\sigma)$. The keys that occur in this pattern are those in the sets $\mathit{recoverable}(M)$ and $\mathit{recoverable}(N\sigma)$, which are therefore equal. Moreover, the value of σ on $\mathit{hidden}(N)$ is irrelevant, since the keys in this set get obliterated with the symbol $\Box$. So, by acyclicity, we can again rename those keys to $K_1, \ldots, K_n$ in such a way that K_I encrypts K_i only if $I \geq i$. This renaming is much as in M, so we omit its justification. We obtain a function σ' and an expression N' such that $N' = N\sigma'$, $M' \equiv N'$, $\mathit{recoverable}(M') = \mathit{recoverable}(N') = \{J_1, \ldots, J_\mu\}$, $\mathit{hidden}(N') = \{K_1, \ldots, K_n\}$, and K_I encrypts K_i in N' only if $I \geq i$.

Continuing our example, let N *be the expression:*

$$\{1\ 1\}_{K_2}\ \{K_3\}_{K_2}\ K_1\ \{K_3\}_{K_2}\ \{K_8\}_{K_2}\ \{1\}_{K_5}\ \{1\ 1\ 1\}_{K_3}\ 0\ \{0\ 0\}_{K_8}\ \{K_3\}_{K_1}$$

so $\mathit{recoverable}(N) = \{K_1, K_3\}$ *and* $\mathit{hidden}(N) = \{K_2, K_5, K_8\}$. *We rename* $(K_1, K_2, K_3, K_5, K_8)$ *to* $(J_1, K_3, J_2, K_1, K_2)$, *obtaining a corresponding expression* N'*:*

$$\{1\ 1\}_{K_3}\ \{J_2\}_{K_3}\ J_1\ \{J_2\}_{K_3}\ \{K_2\}_{K_3}\ \{1\}_{K_1}\ \{1\ 1\ 1\}_{J_2}\ 0\ \{0\ 0\}_{K_2}\ \{J_2\}_{J_1}$$

Note that N' *(and* N*) have a different number of hidden keys than* M' *(and* M*). On the other hand, they have the same number of recoverable keys; this equality is implied by the definition of equivalence.*

In sum, we can thus apply renamings to M and N, obtaining M' and N' such that $pattern(M') = pattern(N')$, M' has hidden keys $K_1, \ldots, K_m$, N' has hidden keys $K_1, \ldots, K_n$, if K_I encrypts K_i then $I \geq i$ in both M' and N', and both have recoverable keys $J_1, \ldots, J_\mu$.

The hybrid patterns M_i and N_j In the next phase of the proof, we introduce patterns (that is, extended expressions) $M_0, M_1, \ldots, M_m$ and $N_0, N_1, \ldots, N_n$ so that these patterns form a chain between M' and N'. Relying on the function p from Section 3.2, we let:

$$M_i \quad = \quad p(M',\ \mathit{recoverable}(M') \cup \{K_1, \ldots, K_i\})$$

where $K_1, \ldots, K_m$ are the hidden keys of M' and $i \in \{0, \ldots, m\}$. In particular, we have $M_0 = pattern(M')$ and $M_m = M'$. Similarly, for $j \in \{0, \ldots, n\}$, we let:

$$N_j \quad = \quad p(N',\ \mathit{recoverable}(N') \cup \{K_1, \ldots, K_j\})$$

and in particular obtain $N_0 = pattern(N')$ and $N_n = N'$. Intuitively, M_i and N_i are the patterns that the adversary sees in M' and N', respectively, if the adversary has a priori knowledge of the otherwise hidden keys $K_1, \ldots, K_i$. The ordering of the keys guarantees that this knowledge does not permit the discovery of other hidden keys.

In our example, we have:

$$
\begin{array}{lcccccccccc}
M' \\
\| \\
M_4: & \{0\}_{K_3} & \{K_1\,1\}_{K_4} & J_1 & \{0\}_{K_2} & \{K_3\}_{K_4} & \{K_1\,K_2\}_{K_4} & \{1\,1\,1\}_{J_2} & 0 & \{K_1\}_{K_3} & \{J_2\}_{J_1} \\
M_3: & \{0\}_{K_3} & \square & J_1 & \{0\}_{K_2} & \square & \square & \{1\,1\,1\}_{J_2} & 0 & \{K_1\}_{K_3} & \{J_2\}_{J_1} \\
M_2: & \square & \square & J_1 & \{0\}_{K_2} & \square & \square & \{1\,1\,1\}_{J_2} & 0 & \square & \{J_2\}_{J_1} \\
M_1: & \square & \square & J_1 & \square & \square & \square & \{1\,1\,1\}_{J_2} & 0 & \square & \{J_2\}_{J_1} \\
M_0: & \square & \square & J_1 & \square & \square & \square & \{1\,1\,1\}_{J_2} & 0 & \square & \{J_2\}_{J_1} \\
\| \\
N_0: & \square & \square & J_1 & \square & \square & \square & \{1\,1\,1\}_{J_2} & 0 & \square & \{J_2\}_{J_1} \\
N_1: & \square & \square & J_1 & \square & \square & \{1\}_{K_1} & \{1\,1\,1\}_{J_2} & 0 & \square & \{J_2\}_{J_1} \\
N_2: & \square & \square & J_1 & \square & \square & \{1\}_{K_1} & \{1\,1\,1\}_{J_2} & 0 & \{0\,0\}_{K_2} & \{J_2\}_{J_1} \\
N_3: & \{1\,1\}_{K_3} & \{J_2\}_{K_3} & J_1 & \{J_2\}_{K_3} & \{K_2\}_{K_3} & \{1\}_{K_1} & \{1\,1\,1\}_{J_2} & 0 & \{0\,0\}_{K_2} & \{J_2\}_{J_1} \\
\| \\
N'
\end{array}
$$

Note that $pattern(M') = M_0 = N_0 = pattern(N')$.

Defining ensembles for the patterns M_i and N_j Next we map each of the patterns $M_0, \ldots, M_m, N_0, \ldots, N_n$ to an ensemble $[\![M_0]\!]_\Pi, \ldots, [\![M_m]\!]_\Pi$, $[\![N_0]\!]_\Pi, \ldots, [\![N_n]\!]_\Pi$, respectively. We define this mapping by extending the conversion algorithm of Figure 1 so that it applies to patterns, not just to expressions. The extension is simple: any time it encounters the symbol $\Box$, it returns the encryption of $\mathbf{0}$ using a new, fixed key, which is used for no other purpose. More precisely, we extend the algorithm of Figure 1 by adding to Initialize the line:

$$\tau(K_0) \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$$

and adding to Convert the lines:

if $M = \Box$ **then**
$\quad y \stackrel{R}{\leftarrow} \mathcal{E}_{\tau(K_0)}(\mathbf{0})$
$\quad$ **return** $\langle\, y$, "ciphertext" $\rangle$

Finding a large gap Clearly $[\![M]\!]_\Pi = [\![M']\!]_\Pi$, since M and M' differ only in their indexing, and similarly $[\![N]\!]_\Pi = [\![N']\!]_\Pi$. Therefore, our goal is to show that $[\![M']\!]_\Pi \approx [\![N']\!]_\Pi$.

We argue by contradiction: we assume that there is an adversary A that distinguishes $[\![M']\!]_\Pi$ and $[\![N']\!]_\Pi$, in order to contradict the type-0 security of Π. According to the definitions, the adversary A runs in polynomial time, and the function:

$$\lambda(\eta) = \Pr[y \stackrel{R}{\leftarrow} [\![M']\!]_\Pi :\ A(\eta, y) = 1] - \Pr[y \stackrel{R}{\leftarrow} [\![N']\!]_\Pi :\ A(\eta, y) = 1]$$

is not negligible, that is, for some constant c, for some infinite set $\mathcal{N}$, $\lambda(\eta) > \eta^{-c}$ for all $\eta \in \mathcal{N}$. For $0 \le i \le m$ and $1 \le j \le n$, we define:

$$\begin{aligned} p_i(\eta) &= \Pr[y \stackrel{R}{\leftarrow} [\![M_i]\!]_{\Pi[\eta]} :\ A(\eta, y) = 1] \\ q_i(\eta) &= \Pr[y \stackrel{R}{\leftarrow} [\![N_j]\!]_{\Pi[\eta]} :\ A(\eta, y) = 1] \end{aligned}$$

Below, we sometimes omit the argument η for notational simplicity. Since $M' = M_m$ and $N' = N_n$, we have that $\lambda = p_m - q_n$. In addition, we have that $p_0 = q_0$ because M' and N' yield the same pattern, so we also have that:

$$\begin{aligned} \lambda \;=\; & (p_m - p_{m-1}) + (p_{m-1} - p_{m-2}) + \ldots + (p_1 - p_0) + \\ & (q_0 - q_1) + (q_1 - q_2) + \ldots + (q_{n-1} - q_n) \end{aligned}$$

We thus have $m+n$ summands that add up to λ. By the triangle inequality, there is either $i \in \{1,\ldots,m\}$ such that $p_i - p_{i-1} \geq \lambda/(m+n)$ or there is $j \in \{1,\ldots,n\}$ such that $q_{i-1} - q_i \geq \lambda/(m+n)$. Moreover, a suitable index i or j exists for each $\eta \in \mathcal{N}$, so there is an index i or j that works for infinitely many $\eta \in \mathcal{N}$, since the number of summands is finite and fixed. Let i be such an index; the case of an index j is exactly analogous. Hence, there exists an infinite set $\mathcal{N}' \subseteq \mathcal{N}$ such that $p_i(\eta) - p_{i-1}(\eta) \geq \lambda(\eta)/(m+n)$ for each $\eta \in \mathcal{N}'$.

In our example, we are assuming that there is some adversary A with a good advantage, say 0.50, in distinguishing $[\![M_4]\!]_{\Pi[\eta]}$ and $[\![N_3]\!]_{\Pi[\eta]}$, that is, $[\![M]\!]_{\Pi[\eta]}$ and $[\![N]\!]_{\Pi[\eta]}$. So the adversary A will distinguish one of the following with advantage at least $0.50/7$: $[\![M_4]\!]_{\Pi[\eta]}$ and $[\![M_3]\!]_{\Pi[\eta]}$; $[\![M_3]\!]_{\Pi[\eta]}$ and $[\![M_2]\!]_{\Pi[\eta]}$; $[\![M_2]\!]_{\Pi[\eta]}$ and $[\![M_1]\!]_{\Pi[\eta]}$; $[\![M_1]\!]_{\Pi[\eta]}$ and $[\![M_0]\!]_{\Pi[\eta]}$; $[\![N_0]\!]_{\Pi[\eta]}$ and $[\![N_1]\!]_{\Pi[\eta]}$; $[\![N_1]\!]_{\Pi[\eta]}$ and $[\![N_2]\!]_{\Pi[\eta]}$; or $[\![N_2]\!]_{\Pi[\eta]}$ and $[\![N_3]\!]_{\Pi[\eta]}$. For example, suppose that it is $[\![M_3]\!]_{\Pi[\eta]}$ and $[\![M_2]\!]_{\Pi[\eta]}$. Then A answers 1 substantially more often when given samples from $[\![M_3]\!]_{\Pi[\eta]}$ than when given samples from $[\![M_2]\!]_{\Pi[\eta]}$.

This step of the proof (finding a large gap) is non-constructive. The non-constructiveness is convenient for our exposition. However, it can be avoided using standard methods, as in [20]: one may simply choose the gap location at random.

Contradicting the type-0 security of Π Using A, we construct an adversary A_0 that violates the type-0 security of Π. The definition of A_0 is in Figure 2. Since A_0 attacks the type-0 security of an encryption scheme, it has access to two oracles, f and g. Those oracles can be instantiated in one of two ways. In one case, the oracle f is $\mathcal{E}_{k_i}(\cdot)$, for a randomly chosen $k_i \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$, while the oracle g is $\mathcal{E}_{k_0}(\cdot)$, for a randomly chosen $k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$. In the other case, the oracle f is $\mathcal{E}_{k_0}(\mathbf{0})$, for a randomly chosen $k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta)$, while the oracle g is again $\mathcal{E}_{k_0}(\mathbf{0})$.

We have:

$$\begin{array}{rcl} p_i(\eta) & = & \Pr[k_i, k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta) :\ A_0^{\mathcal{E}_{k_i}(\cdot),\, \mathcal{E}_{k_0}(\cdot)}(\eta) = 1] \\ p_{i-1}(\eta) & = & \Pr[k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta) :\ A_0^{\mathcal{E}_{k_0}(\mathbf{0}),\, \mathcal{E}_{k_0}(\mathbf{0})}(\eta) = 1] \end{array}$$

These equalities hold because $\textsc{Convert2}(M')$ returns a sample from $[\![M_i]\!]_{\Pi}$ when $f = \mathcal{E}_{k_i}(\cdot)$ and $g = \mathcal{E}_{k_0}(\cdot)$, and $\textsc{Convert2}(M')$ returns a sample from $[\![M_{i-1}]\!]_{\Pi}$ when $f = \mathcal{E}_{k_0}(\mathbf{0})$ and $g = \mathcal{E}_{k_0}(\mathbf{0})$. In both cases, notice that encryption under a recoverable key K corresponds to encryption under the

algorithm $A_0^{f,g}(\eta)$
  for $K \in \mathit{Keys}(M')$ **do** $\tau(K) \xleftarrow{R} \mathcal{K}(\eta)$
  $y \xleftarrow{R}$ CONVERT2(M')
  $b \xleftarrow{R} A(\eta,\ y)$
  return b

algorithm CONVERT2(M^*)
  if $M^* = K$ where $K \in$ **Keys then**
    return $\langle\, \tau(K)$, "key" $\rangle$
  if $M^* = b$ where $b \in$ **Bool then**
    return $\langle\, b$, "bool" $\rangle$
  if $M^* = (M_1^*, M_2^*)$ **then**
    return $\langle$ CONVERT2(M_1^*), CONVERT2(M_2^*), "pair" $\rangle$
  if $M^* = \{M_1^*\}_K$ **then**
    if $K \in \{J_1, \ldots, J_\mu, K_1, \ldots, K_{i-1}\}$ **then**
      $x \xleftarrow{R}$ CONVERT2(M_1^*)
      $y \xleftarrow{R} \mathcal{E}_{\tau(K)}(x)$
      return $\langle\, y$, "ciphertext" $\rangle$
    else if $K = K_i$ **then**
      $x \xleftarrow{R}$ CONVERT2(M_1^*)
      $y \xleftarrow{R} f(x)$
      return $\langle\, y$, "ciphertext" $\rangle$
    else if $K \in \{K_{i+1}, \ldots, K_m\}$ **then**
      $y \xleftarrow{R} g(\mathbf{0})$
      return $\langle\, y$, "ciphertext" $\rangle$

Figure 2: *Given an adversary A that distinguishes $[\![M_i']\!]_\Pi$ from $[\![M_{i-1}']\!]_\Pi$, the adversary A_0 violates the type-0 security of Π, using the oracles f and g. As in the rest of the proof, $K_1, \ldots, K_m$ are the hidden keys and $J_1, \ldots, J_\mu$ the recoverable keys of M'.*

associated key $\tau(K)$. Encryption under a hidden key K in $\{K_1, \ldots, K_{i-1}\}$ also corresponds to encryption under the associated key $\tau(K)$, while encryption under a hidden key in $\{K_{i+1}, \ldots, K_n\}$ results in $\mathbf{0}$ encrypted under k_0. For the first equality, encryption under the hidden key K_i corresponds to encryption under k_i; for the second, encryption under K_i results in $\mathbf{0}$ encrypted under k_0.

We therefore also have:

$$\begin{aligned}
\mathsf{Adv}^0_{\Pi[\eta]}(A_0) &= \Pr[k_i, k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta) : A_0^{\mathcal{E}_{k_i}(\cdot),\, \mathcal{E}_{k_0}(\cdot)}(\eta) = 1] - \\
&\quad \Pr[k_0 \stackrel{R}{\leftarrow} \mathcal{K}(\eta) : A_0^{\mathcal{E}_{k_0}(\mathbf{0}),\, \mathcal{E}_{k_0}(\mathbf{0})}(\eta) = 1] \\
&= p_i(\eta) - p_{i-1}(\eta)
\end{aligned}$$

For infinitely many values of η (those greater than $(m+n)$ in $\mathcal{N}'$), we obtain:

$$\begin{aligned}
\mathsf{Adv}^0_{\Pi[\eta]}(A_0) &\geq \lambda(\eta)/(m+n) \\
&> \eta^{-c}/(m+n) \\
&> \eta^{-(c+1)}
\end{aligned}$$

Hence, the function $\mathsf{Adv}^0_{\Pi[\eta]}(A_0)$ is not negligible. This conclusion contradicts the hypothesis that the encryption scheme Π is type-0 secure, as desired.

Completing our example, let us suppose that A answers 1 substantially more often when given samples from $[\![M_3]\!]_{\Pi[\eta]}$ than when given samples from $[\![M_2]\!]_{\Pi[\eta]}$, as above. We use A to show that the encryption scheme Π is not type-0 secure by constructing a successful adversary A_0 against Π. This adversary relies on two oracles f and g, with two instantiations each. The two instantiations come from the definition of type-0 security. With the first instantiation, A_0 creates a sample from $[\![M_3]\!]_{\Pi[\eta]}$ and then calls A. With the second instantiation, A_0 creates a sample from $[\![M_2]\!]_{\Pi[\eta]}$ and then calls A. Therefore, A_0 answers 1 substantially more often in the first case. $\square$

Theorem 1 gives an asymptotic statement of security. From its proof one can, as always, extract a corresponding concrete-security statement. This statement would say the following. Let M and N be acyclic expressions with m and n keys and lengths $|M|$ and $|N|$, respectively. (The length of an expression is just the number of rules used to generate it.) Suppose that $M \cong N$. Assume further that $m, n \geq 1$, thus excluding only trivial cases. Fix a security parameter η and an encryption scheme Π. Let A be an adversary that runs in time t and achieves advantage $\epsilon(\eta) =$

$\Pr[y \xleftarrow{R} [\![M]\!]_{\Pi[\eta]} : A(\eta, y) = 1] - \Pr[y \xleftarrow{R} [\![N]\!]_{\Pi[\eta]} : A(\eta, y) = 1]$ in distinguishing $[\![M]\!]_{\Pi[\eta]}$ and $[\![N]\!]_{\Pi[\eta]}$. Then there exists an adversary A_0 that breaks the type-0 security of $\Pi[\eta]$ with advantage $\epsilon_0 = \mathsf{Adv}^0_{\Pi[\eta]}(A_0) \geq \epsilon/(m+n)$. Moreover, there exist constants α and α' such that A_0 makes at most $\max\{m, n\}$ queries to its encryption oracles, these queries having length at most $\alpha' \cdot \max\{|M|, |N|\}$; and the running time of A_0 is at most $t+\alpha \cdot T_{\Pi[\eta]} \cdot (|M|+|N|)$, where $T_{\Pi[\eta]}$ is the maximum time to choose a key from $\mathcal{K}(\eta)$ plus the time to encrypt a message of length at most $\alpha' \cdot \max\{|M|, |N|\}$ bits using the key. The numbers α and α' depend only on encoding conventions and details of the model of computation.

One may wonder whether a converse to Theorem 1 holds, that is, whether indistinguishability implies equivalence. Such a converse fails for a fairly trivial reason: if applying the algorithm of Figure 1 to the expressions M and M' gives rise to encryptions of strings outside the message space $\mathsf{Plaintext}$ of the encryption scheme Π, then identical ensembles may be associated with M and M' even when M and M' are not equivalent. We have not explored whether the converse holds when $\mathsf{Plaintext}$ is large enough.

6 Conclusions

The formal approach to cryptography often deals with simple, all-or-nothing assertions about security. The computational approach, on the other hand, makes a delicate use of probability and computational complexity. However, one may intuit that the formal assertions are valid in computational models, if not absolutely at least with high probability and against adversaries of limited computational power. In this paper, we develop this intuition, applying it to the study of encryption. We prove that the intuition is correct under substantial but reasonable hypotheses. This study of encryption is a step—perhaps modest but hopefully suggestive—toward treating security protocols and complete systems, and toward combining the sophistication of computational models with the simplicity and power of formal reasoning.

Acknowledgments

Anand Desai, Andy Gordon, and Jan Jürjens suggested improvements to the presentation of drafts of this paper. Phillip Rogaway was supported in part by NSF CAREER Award CCR-9624560, and by MICRO award 98-129.

References

[1] Martín Abadi. Protection in programming-language translations. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 868–883. Springer-Verlag, July 1998. Also Digital Equipment Corporation Systems Research Center report No. 154, April 1998.

[2] Martín Abadi, Cédric Fournet, and Georges Gonthier. Secure implementation of channel abstractions. In *Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 105–116, June 1998.

[3] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The Spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pages 36–47, 1997.

[4] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, Winter 1998.

[5] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, January 1999. An extended version appeared as Digital Equipment Corporation Systems Research Center report No. 149, January 1998.

[6] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Anonymous encryption. Unpublished manuscript, 2000.

[7] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption: analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, 1997.

[8] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer-Verlag, 1994. To appear in *Journal of Computer and System Sciences*.

[9] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology—CRYPTO '94*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1993.

[10] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[11] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 82)*, pages 112–117, 1982.

[12] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. In *Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 157–166, July 1999.

[13] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989. A preliminary version appeared as Digital Equipment Corporation Systems Research Center report No. 39, February 1989.

[14] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 209–218, 1998.

[15] Richard A. DeMillo, Nancy A. Lynch, and Michael Merritt. Cryptographic protocols. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 383–400, 1982.

[16] Danny Dolev, Shimon Even, and Richard Karp. On the security of ping-pong protocols. In *Advances in Cryptology—CRYPTO '82*, pages 177–186. Plenum Press, 1983.

[17] 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.

[18] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *Proceedings of 24h Annual Symposium on Foundations of Computer Science (FOCS 83)*, pages 34–39, 1983.

[19] Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In *Advances in*

Cryptology—Eurocrypt '99, volume 1592 of *Lecture Notes in Computer Science*, pages 429–444. Springer-Verlag, 1999.

[20] Oded Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, 1993.

[21] Oded Goldreich. Foundations of cryptography (Fragments of a book). Manuscript, on the Web at http://theory.lcs.mit.edu/∼oded, February 1995.

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[23] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[24] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.

[25] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[26] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM Journal on Computing*, 17:281–308, 1988.

[27] James W. Gray, III, Kin Fai Epsilon Ip, and King-Shan Lui. Provable security for cryptographic protocols—exact analysis and engineering applications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 45–58, 1997.

[28] James W. Gray, III and John McLean. Using temporal logic to specify and verify cryptographic protocols (progress report). In *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pages 108–116, 1995.

[29] R. Kemmerer, C. Meadows, and J. Millen. Three system for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, Spring 1994.

[30] Richard A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, May 1989.

[31] 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.

[32] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[33] Nancy Lynch. I/O automaton models and proofs for shared-key communication systems. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 14–29, 1999.

[34] Catherine Meadows. A system for the specification and analysis of key management protocols. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 182–195, 1991.

[35] Jonathan K. Millen, Sidney C. Clark, and Sheryl B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[36] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murϕ. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[37] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.

[38] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Cryptographic security of reactive systems (extended abstract). *Electronic Notes in Theoretical Computer Science*, 32, April 2000.

[39] Steve Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

[40] Paul F. Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28, 1994.

[41] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings 1998 IEEE Symposium on Security and Privacy*, pages 160–171, May 1998.

[42] Dennis Volpano and Geoffrey Smith. Verifying secrets and relative secrecy. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 268–276, 2000.

[43] David Wagner. Re: Security of DES key encrypted with its self???? On the Web at http://www.cs.berkeley.edu/˜daw/my-posts/key-as-iv-broken-again, 1996.

[44] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 82)*, pages 80–91, 1982.