A Computational Interpretation of Dolev-Yao Adversaries

Jonathan Herzog

Laboratory for Computer Science, Massachusetts Institute of Technology

Abstract

The Dolev-Yao model is a simple and useful framework in which to analyze security protocols, but it assumes that the adversary is extremely limited. We show that it is possible for the results of this model to remain valid even if the adversary is given additional power. In particular, we show that there exist situations in which Dolev-Yao adversary can be viewed as a valid abstraction of all realistic adversaries. We do this in a number of steps:

(1) We summarize the strong assumptions placed on the Dolev-Yao adversary as a non-malleability property of public-key encryption in the computational model, an alternate framework with a very powerful adversary.

(2) We re-derive and discuss the indistinguishability property of Abadi and Rogaway [2] in the public-key setting, and show that it is satisfied by computational encryption secure against the chosen-ciphertext attack.

(3) We show that any encryption scheme that satisfies the indistinguishability property also satisfies our (more natural) non-malleability property.

1 Introduction

How can we tell if a cryptographic protocol is secure? Phrased another way, how can we be sure that a given protocol meets a given security goal? Before we can analyze a protocol we need to choose a *model*: a collection of assumptions and proof methods.

The *computational* model, for example, is well known. The messages of a protocol are assumed to be bit-strings from some distribution and the adversary

Email address: `jherzog@mit.edu` (Jonathan Herzog).

is assumed to be an arbitrary algorithm. The cryptographic primitives are assumed to be algorithms (or tuples of algorithms) that satisfy some asymptotic property even in the presence of an arbitrary adversary.

To prove a protocol secure in this model, one would use a *reduction* from the protocol to the underlying primitive. That is, one would show that if there exists an adversary with a significant chance of successfully attacking the protocol, then it can be used to construct an adversary with a significant chance of breaking some cryptographic primitive. If the protocol uses encryption, for example, then one would reduce the security of the protocol to the security of the encryption scheme. Thus, an attack that can successfully break the protocol can be transformed into an attack that can successfully break the encryption scheme. Similarly, if the protocol is based on signatures, the one would show that an attack on the protocol can be transformed into one that can forge a signature. By doing so, one can conclude that if the underlying cryptographic primitives (encryption, signatures) are secure then the protocol must be secure also. (See [5] for an example.)

This is a fairly strong model for analysis. The only assumption placed on the adversary is that it is *efficient*: executing in probabilistic polynomial time (PPT). [1] This assumption is fairly weak, giving the model a solid and meaningful grounding in complexity theory. On the other hand, this model is extremely difficult to use. Reductions tend to be fairly tedious to design, and must be produced 'by hand' for each protocol.

Fortunately, there are alternate models such as the *Dolev-Yao* model [7]. In this setting, messages are assumed to be elements of some abstract algebra, and encryption is an abstract operation on that algebra. The adversary is assumed to be a specific (albeit non-deterministic) state machine, and the only way for the adversary to produce new messages is to perform certain operations on messages it already "knows".

This model has an extremely nice feature: simplicity. Because the computation model is symbolic and the adversary is restricted, it is possible to explicitly represent all of the adversary's possible behaviors in a compact way. General theorems can be proven about the limits of the adversary's powers, and it is relatively easy to show the adversary's goals to be outside its range of possible behaviors. This simplicity allows a great deal of automation. Although the problem of protocol security is undecidable in general [8], it is decidable for an important sub-class of protocols [24]. Furthermore, several automated tools have been successfully used. (See [14,23,26] for typical examples. Also see [16] for a recent survey of the field.)

[1] That is, it has access to an infinite tape of random bits and executes in time polynomial in the length of its (non-random) input.

However, this model also has a drawback: the Dolev-Yao adversary is actually quite weak. Although it can pick from among the allowed operations non-deterministically, the set of allowed operations is fixed and quite small. It is unclear whether security against this restricted adversary implies security against more realistic adversary models. It is also unclear how security statements from the Dolev-Yao model transfer to the computational model.

It seems that one must choose between the simplicity of the Dolev-Yao model and the solid grounding of the computational model. However, is this choice necessary? Are the two models irreconcilable? In particular, is it necessarily true that Dolev-Yao proofs of security will have no computational meaning?

This is a large question, and in this paper we only discuss one small part: the adversary. In particular, we show that the use of sufficiently strong primitives from computational cryptography forces an equivalence of sorts between the Dolev-Yao adversary and all computational adversaries. That is, we do four things:

- We describe the Dolev-Yao model, and extract a natural computational security condition that summarizes its strong assumptions regarding the adversary (Sections 3 and 4).
- We investigate a previous effort in this area [2] that related, in the symmetric-key setting, the passive Dolev-Yao adversary to the passive computational one. In particular, this effort showed that if two Dolev-Yao messages are indistinguishable to the passive Dolev-Yao adversary, then the natural interpretations of these two messages in the computational setting will be indistinguishable to the passive computational adversary. We translate this result to the public-key setting (Section 5).
- We show that this indistinguishability property is no more powerful than other, standard definitions of security from computational cryptography (also Section 5).
- Lastly, we show that our more natural security condition is no more powerful than the indistinguishability property (Section 6).

We finish by discussing avenues for future work (Section 7). First, however, we discuss other efforts in this same area.

2 Related Work

The question we discuss has already been partially addressed in work by Abadi and Rogaway ([1,2], and continued in [10] and [17,18]) which served as a great source of inspiration for this work. In particular, these authors derived and implemented the indistinguishability property that will play such a central role here. The indistinguishability property we define and use in this paper is a direct analogue of theirs, translated from the symmetric-encryption setting to that of asymmetric encryption. Because of differences between these two settings, our indistinguishability property will be stronger than theirs in some places and weaker than theirs in others. More importantly, we will go on to relate our indistinguishability property to the property of *malleability*. That is, we show that the computational adversary can be prohibited from producing any message that could not also be produced by the Dolev-Yao adversary. The relationship between indistinguishability and non-malleability depends on the setting (see [4] for an examination of this issue). In the purely computational setting, for example, non-malleability is strictly stronger than indistinguishability if the computational adversary only has access to the public key. However, non-malleability and indistinguishability are equivalent against the *chosen-ciphertext attack* (i.e, when the adversary has constant access to a decryption oracle, as it will in Definition 12). We will show that indistinguishability implies a weak form on non-malleability in that encryption that satisfies our Dolev-Yao indistinguishability property also satisfies our Dolev-Yao non-malleability property. The converse is commonly true in other settings, and is likely to be true here as well. However, this remains an open question for time being.

Another two related research efforts in this area are those of Backes, Pfitzmann and Waidner [3], and Micciancio and Warinschi [19]. In general, these investigations represent protocol executions in two different ways: a "real" setting and an "ideal" setting. In the "real" setting, the execution of a protocol is represented as the communication of Turing machines that use computational encryption to create bit-string messages. The two lines of research differ in their representation of the "ideal" setting. Backes et. al. use a 'database' that stores all messages and tracks which ones are known by whom. This database allows the adversary to access only those messages it would be able to deduce in the Dolev-Yao paradigm. Micciancio and Warinschi, on the other hand, represent the ideal setting directly as symbolic execution in the Dolev-Yao model. The main results of both efforts state that any behavior that an honest participant can see in the "real" setting could also be seen in the "ideal" setting. Hence, a proof of security in the "ideal" setting will serve as a proof in the "real" setting (modulo negligible probabilities).

These works are extremely compelling. However, they focus attention onto the behavior of the adversary as a whole. That is, they regard the adversary's behavior as an unknowable mystery which cannot be broken into component parts. We, on the other hand, regard the behavior of the adversary as a series of message creations, and leverage a statement about a single creation into a statement about the adversary's behavior as a whole.

A less similar approach to the same problem is a recent effort to incorporate polynomial-time indistinguishability into process algebras [12,13,15,20,21]. Process algebras introduce grammars for processes that typically encompass a large number of higher-level programming constructs. They also introduce a number of algebraic rewrite and cancellation laws that allow one to prove two processes equivalent, that their observable behaviors are equivalent, or that the observable behavior of one process is a subset of the observable behavior of another. In this framework, one can prove a given process to be "safe" by showing that its observable behavior is the same as, or a subset of, the observable behavior of an idealized "specification" process.

This idea has recently been expanded to include new types of "equivalent" behavior. In particular, the definitions of both process and observable behavior have been expanded to include probabilistic behavior. This allows the definition of "observationally equivalent" to mean "indistinguishable to any polynomial-time environment or distinguisher." This approach does not provide the tools necessary to prove an original indistinguishability result, but it does allow one to prove that some given indistinguishability result follows from another one. Furthermore, this derivation uses the high-level rewrite and cancellation rules of the algebra rather than direct reductions.

3 The Dolev-Yao Model

We begin our work by exploring the powers of the adversary in the Dolev-Yao model. There are actually several variations on the Dolev-Yao model, each tailored to a specific tool or application. We provide and discuss a generic example which uses public-key encryption. In this setting, messages are assumed to be elements of an algebra $\mathcal{A}$ of values. There are four types of atomic messages:

- Identifiers (public, predictable, denoted by $\mathcal{I}$)
- Random nonces (private, unpredictable, denoted by $\mathcal{R}$),
- Public keys ($\mathcal{K}_{Pub}$), and
- Private keys ($\mathcal{K}_{Priv}$)

Compound messages are created by two deterministic operations:

- $\mathit{encrypt} : \mathcal{K}_{Pub} \times \mathcal{A} \to \mathcal{A}$
- $\mathit{pair} : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$

We write $\{\!|M|\!\}_K$ for $enc(K, M)$ and $M\,N$ for $pair(M, N)$.[2] We require that there be a bijection

$$inv : \mathcal{K}_{Pub} \to \mathcal{K}_{Priv}$$

and by K^{-1} we mean $inv(K)$ when K is a public key and $inv^{-1}(K)$ when K is a private key.

Although we will consider only public-key encryption in this paper, one could also easily add symmetric encryption to the Dolev-Yao model by introducing a new key type on which the *inv* operation is the identity function.

The algebra is assumed to be *free*: every value has a unique representation.

In the Dolev-Yao model, there are two kinds of active parties: honest participants and the adversary. The honest participants follow the steps of the protocol without deviation. They can engage in multiple runs of the protocol simultaneously and with different parties. Some versions of the model also contain the internal states of honest participants, others do not. We will not consider them in this paper.

The network is assumed to be completely under the control of the adversary, who can record, delete, replay, reroute, reorder, and completely control the scheduling of messages. This is modeled by letting the adversary *be* the network: the honest participants send their messages only to the adversary and receive messages only from the adversary. Thus, we can consider each execution of the protocol to be an alternating sequence of adversary messages ($q_i \in \mathcal{A}$) and environment responses ($r_i \subseteq \mathcal{A}$):

$$r_0\ q_1\ r_1\ q_2\ r_2\ \ldots\ q_{n-1}\ r_{n-1}\ q_n\ r_n$$

Typically, each message or response will be accompanied by such auxiliary information as nominal sender, intended receiver, and so forth. We will ignore this auxiliary information in this work. Such issues as scheduling and routing of messages are considered in the Dolev-Yao analysis of a protocol, which will assume that the adversary can choose the recipient and auxiliary information for its messages with *total* non-determinism. Hence, the Dolev-Yao analysis will capture any strategy chosen by the efficient adversaries of the computational model in choosing the routing information of their messages. In this work, we will focus on the *limited* non-determinism allowed to the Dolev-Yao adversary in choosing its messages' contents.

The main limitation on the choice of message content is that every query q_i must be *derivable* from what is known initially and $r_0, r_1, r_2 \ldots r_{i-1}$. The initial

[2] When three or more terms are written together, such as $M_1\,M_2\,M_3$, we assume they are grouped to the left. That is, $M_1\,M_2\,M_3 = pair(pair(M_1, M_2), M_3)$.

knowledge of the adversary includes at least the following:

(1) the public keys ($\mathcal{K}_{Pub}$),
(2) the private keys of subverted participants ($\mathcal{K}_{Adv} \subseteq \mathcal{K}_{Priv}$),
(3) the identifiers of the principals ($\mathcal{I}$), and
(4) the nonces the adversary itself generates ($\mathcal{R}_{Adv} \subseteq \mathcal{R}$) which are assumed to be distinct from all nonces generated by honest participants.

(Note that the the adversary must receive a set r_0 before it sends its first message. This message can be thought of as an "initialization" from the environment which provides the adversary with any extra information that might be available to it in a particular setting.)

The Dolev-Yao model places severe restrictions on what messages are derivable from others. Analyses in this model tend to focus on the *structure* of protocols. That is, they wish to identify those properties of protocols that exist independently of the encryption schemes used to implement them. Hence, the Dolev-Yao model assumes that the only manipulations the adversary can apply with respect to pairing and encryption are those that *must* be allowed. The pairing operation must allow pairing and separation, and the encryption operator must allow encryption and decryption (with known keys). Thus, for a given message M to be derivable from a set of messages S, it must be possible to produce it by applying the following operations a finite number of times:

- decryption with known or learned private keys,
- encryption with public keys,
- pairing of two known elements, and
- separation of a pair into its components.

To combine these two intuitions:

Definition 1 (Closure) *The* closure of S, *written* $C[S]$, *is the smallest subset of* $\mathcal{A}$ *such that:*

(1) $S \subseteq C[S]$,
(2) $\mathcal{I} \cup \mathcal{K}_{Pub} \cup \mathcal{K}_{Adv} \cup \mathcal{R}_{Adv} \subseteq C[S]$,
(3) If $\{\!|M|\!\}_K \in C[S]$ *and* $K^{-1} \in C[S]$, *then* $M \in C[S]$,
(4) If $M \in C[S]$ *and* $K \in C[S]$, *then* $\{\!|M|\!\}_K \in C[S]$,
(5) If $M\,N \in C[S]$, *then* $M \in C[S]$ *and* $N \in C[S]$, *and*
(6) If $M \in C[S]$ *and* $N \in C[S]$, *then* $M\,N \in C[S]$.

It is the central assumption of the Dolev-Yao model that this closure operation represents the limit of the ability of the adversary to create new messages:

Definition 2 (Dolev-Yao adversary) *Suppose that*

$$r_0\ q_1\ r_1\ q_2\ r_2\ \ldots\ q_{n-1}\ r_{n-1}\ q_n\ r_n$$

is a protocol execution in the Dolev-Yao model, where q_1, q_2,... $q_n \in \mathcal{A}$ are messages from the adversary to honest participants and r_0, r_1,... $r_n \subseteq \mathcal{A}$ are the honest participants' responses. Then for all i, $q_i \in C[r_0 \cup \ldots \cup r_{i-1}]$.

That is, although the Dolev-Yao adversary can choose its messages non-deterministically, it must choose them from within the closure. It is this intuition that we will translate into the computational model.

4 Relating the Dolev-Yao and Computational Messages

In this section, we formalize the intuition of Definition 2 in the language of computational cryptography, using a series of intermediate attempts. Intuitively, we would like to say that it should be hard for the computational adversary to produce a single message outside the closure of its input. Informally:

Attempt 1 *An abstract encryption operator provides* weak[3] Dolev-Yao non-malleability *if $\forall PPT$ adversaries $\mathtt{A}$, $\forall S \subseteq \mathcal{A}$, $\forall M \in (\mathcal{A} \setminus C[S])$*

$$\Pr[N \leftarrow \mathtt{A}(S) : N = M] \text{ is small.}$$

Here, $\Pr[A; B; C : P]$ indicates the probability of predicate P being true after running experiments A, B and C in series. The notation $x \leftarrow D$ indicates x being drawn from distribution D. If D is a set, the uniform distribution is used. If D is an algorithm, we use the distribution over output induced by the distribution of the input and the distribution of D's random coin flips.

Although this attempt contains the desired intuition, there are two small problems:

- It is unclear how a set S of Dolev-Yao messages can be passed as input to a computational adversary, or how a Dolev-Yao message M can be produced as output.
- It is not clear what a "small" probability is.

The purpose of this section is to make the above definition meaningful. Our main tool for doing so will be a mapping from Dolev-Yao messages to their

[3] In Section 7, we will consider stronger formalizations of this same intuition.

computational analogues: probability distributions on bit-strings. The mapping we present here is congruent to that given by Abadi and Rogaway[1,2], adapted to the public-key encryption setting.

The "encoding" of a message M, written $[\![M]\!]^t_\eta$, is a probability distribution that depends on four things:

- The formal message M,
- The *tape* (t) which is an infinite sequence of bits. We assume for convenience that we have random access to this tape, although this can be easily simulated using a standard tape and some book-keeping. In usage, we will assume that the bits on this tape are random.
- A security parameter, which is a natural number η represented in unary. This parameter represents the amount of security present in the system. In encryption schemes, for example, the security parameter can be thought of as the size of keys.
- An arbitrary public-key encryption scheme, which in the computational setting is a triple of algorithms:
  · G is the key generation algorithm, which takes as input a security parameter η, and η bits of randomness. It outputs a public/private key pair.
  · E is the encryption algorithm, which takes in as input a public key, a plaintext string, and η bits of randomness. The output is the ciphertext. We will write $\mathsf{E}(x, pk)$ for the distribution induced by running E (over all choices of randomness) on plaintext x and public key pk.
  · D is the decryption algorithm, which takes as input a private key and a string. It is required that $\mathsf{D}(sk, \mathsf{E}(pk, x, \sigma)) = x$ for all valid key pairs (pk, sk), all plaintexts x, and all choices of randomness $\sigma \in \{0, 1\}^\eta$.

  Note that the key generation and encryption algorithms are randomized. The randomness used by these algorithms will be polynomial in the security parameter η; we assume without loss of generality that all of these polynomials are the identity polynomial $f(\eta) = \eta$.

Definition 3 (Encoding: messages) *Let $\eta \in \mathcal{N}$ be the security parameter. Let $t \in \{0, 1\}^\omega$ be a random tape, partitioned into a length-η segment for each nonce and public key in $\mathcal{A}$. Let $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ be a public-key encryption scheme. Then for any $M \in \mathcal{A}$, the* encoding *of M, written $[\![M]\!]^t_\eta$, is defined recursively as:*

- *If $M \in \mathcal{R}$ is a nonce, then $[\![M]\!]^t_\eta = \langle \sigma_M, \text{"nonce"} \rangle$, where σ_M is the value of the tape partition associated with M.*
- *If (M, M^{-1}) is a public/private key pair, then $[\![M]\!]^t_\eta = \langle e, \text{"pubkey"} \rangle$ and $[\![M^{-1}]\!]^t_\eta = \langle d, \text{"privkey"} \rangle$ where (e, d) is the output of $\mathsf{G}(1^\eta, \sigma_M)$. Note that σ_M is used for randomness.*
- *If $M \in \mathcal{I}$ is an identifier, then $[\![M]\!]^t_\eta$ is mapped to $\langle m, \text{"id"} \rangle$ where m is any (short) bit-string uniquely associated with M. That is, we do not care how*

identifiers are mapped to bit-strings so long as each identifier is uniquely represented. We assume that it is efficient to compute the encoding of a given identifier.

- *If* $M = M_1\, M_2$, *then* $[\![M]\!]^t_\eta$ *is the mapping from pairs of distributions to distributions given by* $\left\langle [\![M_1]\!]^t_\eta, [\![M_2]\!]^t_\eta, \text{“pair”} \right\rangle$.
- *If* $M = \{\!|M'|\!\}_K$ *is an encryption, then* $[\![M]\!]^t_\eta$ *is the mapping from pairs of distributions to distributions given by* $\left\langle \mathsf{E}\left([\![M']\!]^t_\eta, [\![K]\!]^t_\eta\right), [\![K]\!]^t_\eta, \text{“enc”} \right\rangle$

If $S \subseteq \mathcal{A}$, *then by* $[\![S]\!]^t_\eta$ *we mean* $\left\langle [\![s_1]\!]^t_\eta, [\![s_2]\!]^t_\eta, \ldots \right\rangle$ *where* s_1, s_2 *are the elements of* S *in some canonical order. By* $[\![M]\!]_\eta$ *we mean the distribution*

$$\left\{t \leftarrow \{0,1\}^\omega ; m \leftarrow [\![M]\!]^t_\eta : m\right\}.$$

The bits on the tape are used to represent the coin flips used to make atomic elements, and we will later enforce that the tape is filled with random bits. Compound terms are made via either bit-string concatenation or a computational encryption scheme. Note that the coin flips used by the encryption algorithm are *not* taken from the tape. Hence, $[\![\{\!|M'|\!\}_K]\!]^t_\eta$ remains a distribution even if t is fixed.

There are two properties of computational public-key encryption that our encoding mapping will need to accommodate. First, public-key encryption is not required to hide the key used to encrypt. We make this possible leak of information explicit in the definition above by explicitly concatenating each ciphertext with the encrypting key.

Secondly, computational public-key encryption is not generally required to hide the length of the plaintext. For this reason, we need to limit the amount of information about a plaintext that will be revealed by its length. We will assume that the length of a message depends only on the message's structure, not any of its component values. More formally, let the *type tree* of a formal message be the same as its parse tree except that each leaf is replaced by its type. We use the same notation for type trees that we do for messages. Thus, the type tree of a message $\{\!|A\, N|\!\}_K$ (where $A \in \mathcal{I}$, $N \in \mathcal{R}$ and $K \in \mathcal{K}_{Pub}$) is $\{\!|\mathcal{I}\, \mathcal{R}|\!\}_{\mathcal{K}_{Pub}}$.

We assume that the length of a formal message M depends only on $\mathcal{T}_M$, the type tree of M, and the security parameter. This is not an unreasonable assumption. The above definition of the encoding mapping implies that all nonces encode to the same length. The assumption can be trivially enforced for other type trees by padding out to some maximal length. Thus, we will use $\left|[\![M]\!]^t_\eta\right|$ to designate the unique length of encodings of M.

The encoding mapping allows formal messages to be represented as bit-strings, which allows formal messages to be passed to and returned by the computational adversary. This solves the first problem with Attempt 1. Because the mapping also introduced the security parameter, we can solve the second problem. A probability is "small" if it is *negligible* in the security parameter:

Definition 4 (Negligible) *A function $f : \mathcal{N} \to \mathcal{R}$ is* negligible *if, for any polynomial q, $f(\eta) \leq \frac{1}{q(\eta)}$ for all sufficiently large η.*

(The phrase "for all sufficiently large η" is equivalent to $\exists \eta_0 .\, \forall \eta \geq \eta_0$.)

With these two problems solved, we can re-attempt to translate Definition 2 into computational terms:

Attempt 2 *An encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ provides* weak Dolev-Yao non-malleability *if, when used in $[\![\cdot]\!]^t_\eta$,*

$$
\begin{array}{l}
\forall\, PPT \text{ adversaries } \mathtt{A},\ \forall\, S \subseteq \mathcal{A},\ \forall M \in (\mathcal{A} \setminus C[S]) \\
\forall\, \text{polynomials } q,\ \forall \text{ sufficiently large } \eta : \\
\Pr[\ t \leftarrow \{0,1\}^\omega ; \\
\quad s \leftarrow [\![S \cup \mathcal{K}_{Pub} \cup \mathcal{K}_{Adv} \cup \mathcal{R}_{Adv} \cup \mathcal{I}]\!]^t_\eta ; \\
\quad m \leftarrow \mathtt{A}(1^\eta, s) : \\
\quad m \in supp\ [\![M]\!]^t_\eta \qquad\qquad\qquad\qquad] \leq \frac{1}{q(\eta)}
\end{array}
$$

Here, $supp\, D$ means the support of distribution D. When the support of a distribution contains one element, we will treat the distribution itself as a singleton set.

This definition is still problematic, however, for two technical reasons. First, the input to the adversary might be of infinite length. The set S may be of infinite length. There may be an infinite number of elements in $\mathcal{I}$, $\mathcal{R}_{Adv}$, $\mathcal{K}_{Pub}$ and $\mathcal{K}_{Adv}$. If any of these are the case, then the restriction of the adversary to probabilistic polynomial-time is meaningless. No computational encryption scheme would remain secure against an infinite-time adversary. For this reason, we require that S be of finite size. The sets $\mathcal{I}$, $\mathcal{R}_{Adv}$, $\mathcal{K}_{Pub}$ and $\mathcal{K}_{Adv}$ might still be infinite, so instead of passing them as input we represent them via oracles:

- $\mathtt{M}^t_\eta(x)$ returns (the encoding of) the identifier of the xth participant.
- $\mathtt{R}^t_\eta(x)$ returns the (encoding of) the xth nonce in $\mathcal{R}_{Adv}$,
- $\mathtt{PbK}^t_\eta(x)$ returns the public key of principal x, and
- $\mathtt{PrK}^t_\eta(x)$ returns the private key of x of $x \in [\![K^{-1}]\!]^t_\eta$ if $K^{-1} \in \mathcal{K}_{Adv}$.

The second problem is that our results rely upon a technical limitation: acyclicity of encryptions. A set of encryptions is acyclic if, when K_1 encrypts K_2^{-1} in some element of S, and K_2 encrypts K_3^{-1}, and so on, this sequence of keys encrypting keys never loops back on itself. More formally:

Definition 5 (Acyclic) *For an expression M, construct a graph G_M where the nodes are the public/private key pairs used in the expression. We draw an edge from $p_1 \rightarrow p_2$ if in M the private key K_2^{-1} associated with pair p_2 is encrypted with K_1, the public key associated with p_1. The expression M is* acyclic *if the graph G_M is acyclic.*

Our results will only hold for acyclic sets S. However, protocols analyzed in the Dolev-Yao model typically operate in one of three ways:

- Long-term keys are used to encrypt session keys, which themselves never encrypt other keys,
- The present session key is used to encrypt the next session key, but never the previous, or
- Keys are never encrypted at all.

None of these cases will produce cyclic encryptions.

Thus, we arrive at our final security condition:

Definition 6 (Dolev-Yao weak non-malleability) *An encryption scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *provides* weak Dolev-Yao non-malleability *if, when used in* $[\![\cdot]\!]_\eta^t$,

$$
\begin{aligned}
&\forall\, PPT \textit{ adversaries } \mathtt{A},\ \forall \textit{ acyclic finite } S \subseteq \mathcal{A},\ \forall M \notin C[S]\,, \\
&\forall \textit{ polynomials } q,\ \forall \textit{ sufficiently large } \eta : \\
&\Pr[\ t \leftarrow \{0,1\}^\omega \\
&\quad s \leftarrow [\![S]\!]_\eta^t\,; \\
&\quad m \leftarrow \mathtt{A}^{\mathcal{M}_\eta^t(\cdot), \mathcal{P}b\mathcal{K}_\eta^t(\cdot), \mathcal{P}r\mathcal{K}_\eta^t(\cdot), \mathcal{R}_\eta^t(\cdot)}(1^\eta, s) : \\
&\quad m \in supp\ [\![M]\!]_\eta^t \qquad\qquad\qquad] \leq \tfrac{1}{q(\eta)}
\end{aligned}
$$

The main purpose of this section has been to derive this security condition, which directly captures the assumptions of the Dolev-Yao adversary. However, there exist other security conditions that formalize the Dolev-Yao model. We consider one of these in the next section.

5 An Indistinguishability Lemma

In this section, we consider the indistinguishability-based definitions of Dolev-Yao security originally derived by Abadi and Rogaway [1,2]. Intuitively, the definition of that paper describes when two formal messages should "look" the same to the formal adversary. A formal adversary has the power to make certain, limited deductions from formal messages; two given formal messages should "look" the same when all possible deductions that can be made about them yield the same results. In particular, the formal adversary of [1,2] is assumed to be unable to distinguish between two different encryptions (unless it has the corresponding private key or keys). For example, if the adversary of [1,2] has no other information, the two messages

$$\left\{\!\left|\, \{\!|A|\!\}_{K_2}\ B \,\right|\!\right\}_{K_1}\ K_1^{-1} \quad \text{and} \quad \left\{\!\left|\, \{\!|C\, D|\!\}_{K_3}\ B \,\right|\!\right\}_{K_1}\ K_1^{-1}$$

should be indistinguishable to it no matter what A, B, C and D are.

The fundamental result of Abadi and Rogaway is that if the encoding algorithm uses sufficiently strong computational encryption, then two messages indistinguishable to the formal adversary will encode to distributions indistinguishable to the computational adversary. Their result applies to the case of symmetric encryption, and we will here translate it to the case of public-key encryption. This translation will simultaneously strengthen and weaken the result. Indistinguishability in the public-key setting requires a stronger similarity between messages than was necessary in the case of symmetric encryption. However, our results will be able to tolerate the presence of a previously-absent strong decryption oracle.

Let T be a set of keys and suppose that the formal adversary can decrypt with regard to them. Then we represent the information that such an adversary can deduce from a formal message by its *public-key pattern* [4]:

Definition 7 (Public-key pattern) *Let $T \subseteq \mathcal{K}_{Pub}$. We recursively define the function $p(M,T)$ to be:*

- $p(K,T) = K$ *if* $K \in \mathcal{K}$
- $p(A,T) = A$ *if* $A \in \mathcal{I}$
- $p(N,T) = N$ *if* $N \in \mathcal{R}$
- $p(N_1\, N_2, T) = p(N_1,T)\, p(N_2,T)$
- $p(\{\!|M|\!\}_K, T) = \begin{cases} \{\!|p(M,T)|\!\}_K & \textit{if } K \in T \\ \langle\!|\mathcal{T}_M|\!\rangle_K & \textit{o.w. (where } \mathcal{T}_M \textit{ is the type tree of } M\textit{)} \end{cases}$

[4] We will use "pattern" to indicate public-key pattern, as opposed to the stronger, symmetric-key definition of "pattern" in [2].

Then $pattern_{pk}(M, T)$, *the* public-key pattern *of an expression* M *relative to the set* T, *is*

$$p(M, \mathcal{K}_{Pub} \cap C[\{M\} \cup T]).$$

If $S \subseteq \mathcal{A}$ *is a set of messages, then* $pattern_{pk}(S, T)$ *is*

$$\langle p(s_1, C[S \cup T]), p(s_2, C[S \cup T]), \ldots \rangle$$

where s_1, s_2, ... *are the elements of* S *is some canonical order. The* base pattern *of a message* M, *denoted* $pattern_{pk}(M)$, *is defined to be* $pattern_{pk}(M, \emptyset)$, *and* $pattern_{pk}(S)$ *is defined to be* $pattern_{pk}(S, \emptyset)$.

The grammar/algebra for patterns is exactly that of messages, with the addition of a new kind of leaf node: $\langle\!|\mathcal{T}_M|\!\rangle_K$ (a "blob" of type-tree $\mathcal{T}_M$ under key K) which represents undecipherable encryptions. Unlike the "blobs" of the symmetric-encryption patterns of [1,2], these "blobs" are labeled with K and $\mathcal{T}_M$. This is because computational encryption schemes do not necessarily hide either the encrypting key or the plaintext length.

For convenience, we define a useful relationship between two patterns:

Definition 8 (Ingredient) *If* M, M' *are two patterns, then* M *is an* ingredient *of* M', *written* $M \sqsubseteq M'$, *if the parse tree of* M *is a sub-tree of the parse tree of* M'.

We note that since messages are special forms of patterns, this relationship can be applied between two messages as well as between a message and a pattern. We also note a relationship between a message and its pattern:

Theorem 9 *If* M, M' *are messages and* $M' \sqsubseteq pattern_{pk}(M)$, *then* $M' \in C[M]$.

PROOF. Suppose that $M' \sqsubseteq pattern_{pk}(M)$. Consider the same path from root to M' in the parse tree of M. Along this path, if an interior node (not itself M') is in $C[M]$ then both child nodes are in $C[M]$:

- $C[M]$ is closed under separation. Hence, if a node is the pair $N\,N'$ and the node is in $C[M]$, then both N and N' are in $C[M]$.
- The two children of a node $\{\!|N|\!\}_K$ are N and K. Since $K \in \mathcal{K}_{Pub}$, $K \in C[M]$ automatically. Furthermore, $K^{-1} \in C[M]$ as well: if it were not, then this node of M's parse tree would have been replaced with $\langle\!|\mathcal{T}_N|\!\rangle_K$ in the parse tree of $pattern_{pk}(M)$. But $\langle\!|\mathcal{T}_N|\!\rangle_K$ is not a message and will not contain M' in its parse tree. So $K^{-1} \in C[M]$, and since $C[M]$ is closed under decryption with keys it contains, $N \in C[M]$.

Since the root of this path, M itself, is in $C[M]$ by definition, it must be the case that every child of every node above M' in the parse tree of M is in the set $C[M]$. Hence, $M' \in C[M]$ as well. ■

We can extend the encoding operation to the pattern algebra:

Definition 10 (Encoding: patterns) *Let:*

- $[\![\langle\!| M |\!\rangle]\!]^t_\eta$ *be any fixed bit-string of length* $\left|[\![M]\!]^t_\eta\right|$ *such as the all-zero string, and*
- $[\![\langle\!| M |\!\rangle_K]\!]^t_\eta$ *be the the mapping from distributions to distributions given by*

$$\left\langle \mathsf{E}\left([\![\langle\!| M |\!\rangle]\!]^t_\eta, [\![K]\!]^t_\eta\right), [\![K]\!]^t_\eta, \text{“enc”} \right\rangle .$$

Patterns allow us to state when two messages appear to be the same to the formal adversary: when they have the same pattern. The standard definition of ‘appears to be the same’ in the world of computational encryption is that of *computational indistinguishability.* We present a more general definition, which incorporates the possibility of an oracle:

Definition 11 (Computational indistinguishability) *Suppose that* $\{D_\eta\}_\eta$ *and* $\left\{D'_\eta\right\}_\eta$ *are two families of distributions indexed by the security parameter. Then they are* computationally indistinguishable with respect to a family of oracles $\mathtt{O}_x$, *written* $D_\eta \cong_{\mathtt{O}_x} D'_\eta$, *if*

$$\begin{aligned} &\forall\, PPT \text{ adversaries } \mathtt{A},\ \forall \text{ polynomials } q,\ \forall \text{ sufficiently large } \eta :\\ &\left|\Pr[d \leftarrow D_\eta : 1 \leftarrow \mathtt{A}^{\mathtt{O}_d(\cdot)}(d,\eta)] - \Pr[d \leftarrow D'_\eta : 1 \leftarrow \mathtt{A}^{\mathtt{O}_d(\cdot)}(d,\eta)]\right| \leq \frac{1}{q(\eta)} \end{aligned}$$

We note that if no oracle access is granted at all, then the above definition reduces to the standard notion of computational indistinguishability.

Our intuitive notion is that a message and its pattern should appear to be the same. We formalize this notion by saying that a message and its pattern should encode to computationally indistinguishable probability distributions. To make this formalization completely meaningful, however, we must consider what oracle (if any) the adversary can access. This will be determined by the oracles allowed by the underlying computational encryption scheme.

A computational public-key encryption scheme provides *indistinguishability against the chosen-ciphertext attack* [5] (also written *CCA-2 secure* in the notation of [4]) if no adversary has a chance significantly better than random of

[5] See [25], which builds on the work of [22]. See also [6] for a practical implementation.

determining accurately whether a ciphertext c is the encryption of message m_0 or message m_1, even if:

- the adversary chooses m_0 and m_1 itself, after seeing the given public key, and
- the adversary can access a decryption oracle both before choosing the messages and after receiving the ciphertext in question. (The decryption oracle will not decrypt c itself, however.)

More formally:

Definition 12 (Chosen-ciphertext security) *A computational public-key encryption scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *provides* indistinguishability under the chosen-ciphertext attack *if*

$$
\begin{aligned}
&\forall\, PPT \text{ adversaries } \mathtt{A},\ \forall \text{ polynomials } q,\ \forall \text{ sufficiently large } \eta :\\
&\Pr[\ (pk, sk) \leftarrow \mathsf{G}(1^\eta);\\
&\quad m_0, m_1 \leftarrow \mathtt{A}^{\mathsf{D}_1(\cdot)}(pk);\\
&\quad i \leftarrow \{0,1\}\,;\\
&\quad c \leftarrow \mathsf{E}(m_i, pk);\\
&\quad g \leftarrow \mathtt{A}^{\mathsf{D}_2(\cdot)}(c)\ :\\
&\quad b = g \qquad\qquad\qquad\qquad] \leq \tfrac{1}{2} + \tfrac{1}{q(\eta)}
\end{aligned}
$$

The oracle $\mathsf{D}_1(x)$ *returns* $\mathsf{D}(x, sk)$*, and* $\mathsf{D}_2(x)$ *returns* $\mathsf{D}(x, sk)$ *if* $x \neq c$ *and returns* $\perp$ *otherwise. The adversary is assumed to keep state between the two invocations. It is required that* m_0 *and* m_1 *be of the same length.*

In the terminology of [1,2], this definition requires that encryption be message-hiding. It does not, on the other hand, require that it be key-hiding or length-hiding. It is for this reason that "blobs" in Definition 7 are labeled with both encrypting key and type-tree (which indicates length of plaintext).

We will assume that the encoding mapping uses CCA-2 secure cryptography. Thus, the oracle we will use in Definition 11—to show that a message and its pattern produce indistinguishable encodings—will exactly mirror the decryption oracles of Definition 12. Those oracles will decrypt, with respect to a given public key, anything but a given "challenge" ciphertext. Our oracles will do the same. However, a message and its pattern can be thought of as possibly many different "challenge" ciphertexts under possibly many different keys. It is simple to define the keys with respect to which our oracles will decrypt:

Definition 13 *Let* M *be a pattern. Then* $M|_{\mathcal{K}_{Pub}} = \{K \in \mathcal{K}_{Pub} : K \sqsubseteq M\}$*. If* S *is a set of messages, then* $S|_{\mathcal{K}_{Pub}} = \{K \in \mathcal{K}_{Pub} : \exists M \in S \text{ s. t. } K \sqsubseteq M\}$*.*

In addition, the oracle may decrypt with respect to additional keys in some set T. (We use this additional flexibility in the proof of our main theorem.) Due to efficiency concerns, however, the set T must be finite.

It is more difficult to define the "challenge" ciphertexts which our oracle will not decrypt. Most directly, they are those encryptions which differ between $[\![M]\!]_\eta^t$ and $\left[\!\left[pattern_{pk}\left(M, T\right)\right]\!\right]_\eta^t$. That is, the challenge ciphertexts should be those which correspond to "blobs" in the pattern of M relative to the set of keys T. However, for convenience, we will define a larger but equivalent set of challenge ciphertexts which correspond not only to the "blobs" but all encryptions visible in M to a Dolev-Yao adversary.

Definition 14 (Visible) *Let σ be a bit-string, and τ a set of computational public keys. Then let $vis_\tau\left(\sigma\right)$ be the smallest set so that*

- $\sigma \in vis_\tau\left(\sigma\right)$,
- *if* $\langle a, b, \text{"pair"}\rangle \in vis_\tau\left(\sigma\right)$, *then* $a \in vis_\tau\left(\sigma\right)$ *and* $b \in vis_\tau\left(\sigma\right)$,
- *if* $\langle c, k, \text{"enc"}\rangle \in vis_\tau\left(\sigma\right)$, $k \in \tau$, *and* k' *is the secret key corresponding to* k, *then* $\mathsf{D}(c, k') \in vis_\tau\left(\sigma\right)$, *and*
- *if* $\langle c, k, \text{"enc"}\rangle \in vis_\tau\left(\sigma\right)$, $\langle k', \text{"privkey"}\rangle \in vis_\tau\left(\sigma\right)$, *and* k' *is the secret key corresponding to* k, *then* $\mathsf{D}(c, k') \in vis_\tau\left(\sigma\right)$.

A bit-string m is a visible element *in σ relative to τ if $m \in vis_\tau\left(\sigma\right)$.*

Intuitively, $x \in vis_\tau\left(\sigma\right)$ iff x is an encoding of X, σ is an encoding of M, τ is an encoding of T and $X \sqsubseteq pattern_{pk}\left(M, T\right)$. That is, a bit-string is a visible element of σ if the adversary can derive it from σ using only Dolev-Yao-style operations using σ and keys in τ. The set $vis_\tau\left(\sigma\right)$ contains every ciphertext which corresponds to a "blob" in $pattern_{pk}\left(M, T\right)$. However, it also contains every other ciphertext that has an corresponding analogue in $pattern_{pk}\left(M, T\right)$. The decryption oracle will not decrypt these, but this not worrisome: the computational adversary can decrypt these "non-blobs" itself. Just as these encryptions are not "blobbed" in $pattern_{pk}\left(M, T\right)$ because the required formal private key is in T or derivable from M, the adversary can decrypt the corresponding computational ciphertext from keys in τ or derivable from σ itself. Thus, we can prohibit the decryption of this more general set without losing generality.

Now that we know the nature of our decryption oracle, we can finally define our indistinguishability property between messages and their patterns:

Definition 15 (Dolev-Yao public-key indistinguishability) *A computational encryption scheme provides* Dolev-Yao public-key indistinguishability *if, when used in* $[\![\cdot]\!]_\eta^t$*, for all acyclic formal messages* M *and finite* $T \subseteq \mathcal{K}_{Pub}$*:*

$$[\![M]\!]_\eta \cong_{\mathsf{O}_x^{M,T}} \left[\!\!\left[pattern_{pk}\left(M, T\right) \right]\!\!\right]_\eta$$

where $\mathsf{O}_d^{T,M}(x, pk)$ *returns* $\perp$ *unless* pk *is a valid public key and*

- *either* $pk \in [\![K]\!]_\eta^t$ *for some* $K \in T$*, or*
- $pk \in [\![K]\!]_\eta^t$ *for some* $K \in (M|_{\mathcal{K}_{Pub}} \setminus T)$ *and* x *is not in* $vis_{[\![T]\!]_\eta^t}(d)$*.*

(The tape t *is assumed to be consistent with that used to form the sample from* $[\![M]\!]_\eta$ *or* $\left[\!\!\left[pattern_{pk}\left(M, T\right)\right]\!\!\right]_\eta$*.) In these cases,* $\mathsf{O}_d^{T,M}(x, pk)$ *returns* $\mathsf{D}(x, sk)$ *where* sk *is the private key corresponding to* pk*.*

In the next section, we will show that Dolev-Yao public-key indistinguishability implies Dolev-Yao weak non-malleability. Before this, however, we show that Dolev-Yao public-key indistinguishability can be satisfied by CCA-2 security.

Theorem 16 *If* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *provides indistinguishability under the chosen-ciphertext attack, then* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *provides Dolev-Yao public-key indistinguishability.*

PROOF.

Suppose that the encoding mapping uses a computational encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$. Further, suppose that there exists a formal message M, a set of keys T and a PPT adversary $\mathtt{A}$ that can distinguish between a sample from $[\![M]\!]_\eta^t$ and a sample from $\left[\!\!\left[pattern_{pk}\left(M, T\right)\right]\!\!\right]_\eta^t$ (given access to the oracle in Definition 15). Then $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ does not satisfy CCA-2 security.

We prove this by hybrid argument. Since M is acyclic, we can order the key-pairs used in the parse tree of M as K_1, $K_2 \ldots K_k$ so that if $K_i \rightarrow K_j$ in the graph G_M, then $i \geq j$. That is, the deeper the key in the encryptions, the smaller the number.

We go about the hybrid argument by constructing a number of intermediate patterns between M and $pattern_{pk}\left(M, T\right)$. In particular, we construct patterns M_0, M_1,$\ldots M_k$ such that:

- $M_0 = M = pattern_{pk}\left(M, T \cup \left\{K_1^{-1}, K_2^{-1}, \ldots K_k^{-1}\right\}\right)$,
- $M_i = pattern_{pk}\left(M, T \cup \left\{K_{i-1}^{-1}, K_{i-2}^{-1}, \ldots K_k^{-1}\right\}\right)$, and
- $M_k = pattern_{pk}\left(M, T\right)$.

That is, between M_i and M_{i+1} we pick a key K and replace all encryptions with that key with blobs of the appropriate length.

We use this typeface for a running example. Suppose

$$M = \{\!|A|\!\}_{K_1} \left\{\!\left|K_1^{-1}\right|\!\right\}_{K_2} \{\!|B|\!\}_{K_3} \{\!|A\,B|\!\}_{K_2}$$

and

$$T = \{K_3, K_4\}$$

Assume for now that $\mathcal{K}_{Adv} = \emptyset$. The pattern of M is

$$pattern_{pk}(M,T) = \langle\!|\mathcal{I}|\!\rangle_{K_1} \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2} \{\!|B|\!\}_{K_3} \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2}$$

By using the order on keys suggested by the notation, we can let

$$\begin{aligned}
M_0 = M &= \{\!|A|\!\}_{K_1} \left\{\!\left|K_1^{-1}\right|\!\right\}_{K_2} \{\!|B|\!\}_{K_3} \{\!|A\,B|\!\}_{K_2} \\
M_1 &= \langle\!|\mathcal{I}|\!\rangle_{K_1} \left\{\!\left|K_1^{-1}\right|\!\right\}_{K_2} \{\!|B|\!\}_{K_3} \{\!|A\,B|\!\}_{K_2} \\
M_2 &= \langle\!|\mathcal{I}|\!\rangle_{K_1} \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2} \{\!|B|\!\}_{K_3} \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2} \\
M_3 &= \langle\!|\mathcal{I}|\!\rangle_{K_1} \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2} \{\!|B|\!\}_{K_3} \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2} \\
M_4 &= \langle\!|\mathcal{I}|\!\rangle_{K_1} \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2} \{\!|B|\!\}_{K_3} \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2}
\end{aligned}$$

We will use the hybrid argument on this table.

Now, suppose that the distributions $[\![M]\!]_\eta$ and $\left[\!\left[pattern_{pk}(M,T)\right]\!\right]_\eta$—the top and bottoms rows of our table—are distinguishable. That is, $[\![M]\!]_\eta \not\approx_{\mathsf{O}_x^{M,T}} \left[\!\left[pattern_{pk}(M,T)\right]\!\right]_\eta$. Then we know by a (standard) hybrid argument that two consecutive rows are also distinguishable.[6] We continue the hybrid argument by creating a new table between the two distinguishable rows. Suppose that K_i is the key being "blobbed" between the two rows. Then there are a fixed number of encryptions being converted to "blobs". Create a row for each such encryption, so that two consecutive rows differ only in a single encryption being replaced with a blob.

For example, if the two rows are

$$M_1 = \langle\!|\mathcal{I}|\!\rangle_{K_1} \left\{\!\left|K_1^{-1}\right|\!\right\}_{K_2} \{\!|B|\!\}_{K_3} \{\!|A\,B|\!\}_{K_2}$$

and

$$M_2 = \langle\!|\mathcal{I}|\!\rangle_{K_1} \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2} \{\!|B|\!\}_{K_3} \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2}$$

[6] This only follows if the number of rows in the table is polynomial in the security parameter. In this case, however, the number of rows in the table is constant with respect to η.

Then we could expand this into the table:

$$\begin{aligned} M_1 &= \langle\!|\mathcal{I}|\!\rangle_{K_1}\ \left\{\!\left|K_1^{-1}\right|\!\right\}_{K_2}\ \{\!|B|\!\}_{K_3}\ \{\!|A\,B|\!\}_{K_2} \\ M_{1.5} &= \langle\!|\mathcal{I}|\!\rangle_{K_1}\ \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2}\ \{\!|B|\!\}_{K_3}\ \{\!|A\,B|\!\}_{K_2} \\ M_2 &= \langle\!|\mathcal{I}|\!\rangle_{K_1}\ \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2}\ \{\!|B|\!\}_{K_3}\ \langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2} \end{aligned}$$

Two of these rows must be distinguishable.

Again, there must exist two consecutive rows R_1 and R_2 that can be distinguished. Since the rows differ only in the contents of a single encryption and every other part of the row can be created independently, distinguishing between the encoding of two rows reduces to distinguishing between two encryptions.

Let $\mathtt{A}$ be the adversary that can distinguish between the two rows, and let $E = \{\!|P|\!\}_K$ be the encryption that is being changed into $\langle\!|\mathcal{T}_P|\!\rangle_K$. Then to break the CCA-2 security of the encryption scheme we will distinguish between an encryption of m_0 and m_1 under public key pk by:

- letting $m_0 \leftarrow [\![P]\!]_\eta^t$,
- letting $m_1 \leftarrow [\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$, and
- treating pk as the encoding of K.

More formally:

- On input pk, select random $t \leftarrow \{0,1\}^\omega$. Then draw $p \leftarrow [\![P]\!]_\eta^t\,[pk/K]$, where $[\![M]\!]_\eta^t\,[x/X, y/Y, \ldots]$ is the same as $[\![M]\!]_\eta^t$ except that x is assumed to be the value for X, y the value for Y, and so on. (If X is an encryption and occurs more than once, then x is used as the value for the instance of X indicated by context. Values for the other instances are still drawn as before.) Note that because M is acyclic, we do not need to know the value $[\![K^{-1}]\!]_\eta^t$ to draw from $[\![P]\!]_\eta^t$. Return p and $[\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$ as candidate plaintexts. (Recall that $[\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$ is a fixed string of the appropriate length, such as the all-zero string.)
- On input c, an encryption of either $[\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$ or p, sample $s \leftarrow [\![R_1]\!]_\eta^t\,[c/P, pk/K]$. Note that, since both t and pk were selected randomly, $[\![R_1]\!]_\eta^t\,[c/P, pk/K]$ is the same distribution as $[\![R_1]\!]_\eta$ if c encrypts p. Similarly, $[\![R_1]\!]_\eta^t\,[c/P, pk/K]$ is the same distribution as $[\![R_2]\!]_\eta$ if c encrypts $[\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$. Feed $(s, 1^\eta)$ to $\mathtt{A}$.
- If $\mathtt{A}$ makes an oracle call on (x, pk), we check that $pk = [\![K_0]\!]_\eta^t$ for some $K_0 \in M|_{\mathcal{K}_{Pub}} \cup T$. If not, we return $\perp$. If so, we decrypt or not as follows:
  - · If $K_0 = K$, we check that x is not visible in x relative to $t = [\![T]\!]_\eta^t$. Since x is not visible in s relative to t, and c is visible in s relative to t, $x \neq c$. Hence, the decryption oracle D_2 in Definition 12 will happily decrypt x for us.

- If $K_0 \neq K$, we can produce $\left[K_0^{-1}\right]_\eta^t$ ourselves from the tape t. If $K_0 \in T$, we decrypt x with the value so produced. If $K_0 \in M|_{\mathcal{K}_{Pub}} \setminus T$, we also check to see if x is visible in s relative to t. We return $\perp$ if it is, and decrypt x if it is not.

Assume in our example that rows $M_{1.5}$ and M_2 can be distinguished by `A`*. Then $P = A\,B$ and $K = K_2$. We build the two candidate ciphertexts by selecting $t \leftarrow \{0,1\}^\omega$. We then select $p \leftarrow [\![A\,B]\!]_\eta^t$, and return p, $[\![\langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle]\!]_\eta^t$ as candidate ciphertexts. When we get c, a value either from $\left[\!\!\left[\langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle_{K_2}\right]\!\!\right]_\eta^t$ or $\left[\!\!\left[\{\!|A\,B|\!\}_{K_2}\right]\!\!\right]_\eta^t$, we draw*

$$s \leftarrow \left[\!\!\left[\langle\!|\mathcal{I}|\!\rangle_{K_1}\ \langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2}\ \{\!|B|\!\}_{K_3}\ \{\!|A\,B|\!\}_{K_2}\right]\!\!\right]_\eta^t [c/\ \{\!|A\,B|\!\}_{K_2}, pk/K_2]$$

Since either $s \in supp\ [\![M_{1.5}]\!]_\eta^t$ or $s \in supp\ [\![M_2]\!]_\eta^t$, the adversary `A` *will tell us which one, and this answer will tell us if c encrypts $[\![A\,B]\!]_\eta^t$ or $[\![\langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle]\!]_\eta^t$.*

We simulate `A` *on s: when* `A` *requests that we decrypt a string x with $\left[\!\!\left[K_1^{-1}\right]\!\!\right]_\eta^t$, we make sure that it isn't c, $[\![B]\!]_\eta^t$, or the bit-strings in s that represent $\langle\!|\mathcal{I}|\!\rangle_{K_1}$ and $\langle\!|\mathcal{K}_{Priv}|\!\rangle_{K_2}$. If it is not these four things, we use the tape t to create the secret key and decrypt x.If* `A` *asks us to decrypt something with $\left[\!\!\left[K_2^{-1}\right]\!\!\right]_\eta^t$, we check that it is not any of the four ciphertexts above. If it is not, then we send it to the decryption oracle provided to us in Definition 12, which will decrypt it for us. If* `A` *asks us to decrypt with $\left[\!\!\left[K_3^{-1}\right]\!\!\right]_\eta^t$ or $\left[\!\!\left[K_4^{-1}\right]\!\!\right]_\eta^t$, we create the keys from the tape and decrypt any ciphertext.*

The answer from `A` *directly corresponds to the plaintext chosen for c, which allows us to distinguish whether it encrypts $[\![\langle\!|\mathcal{I}\,\mathcal{I}|\!\rangle]\!]_\eta^t$ or p.*

`A`(s, η) will eventually return an answer that distinguishes between samples from R_1 and R_2. The answer from `A` will signify whether c encrypted p or $[\![\langle\!|\mathcal{T}_P|\!\rangle]\!]_\eta^t$. ■

We note as a corollary that the exact analogue of the Abadi-Rogaway result holds: if two messages M and N have the same pattern (with respect to some set T) then they produce indistinguishable encodings:

Corollary 17 *Suppose that M, N are two acyclic messages, $T \subseteq \mathcal{A}$ is a set of keys, and $M|_{\mathcal{K}_{Pub}} = N|_{\mathcal{K}_{Pub}}$. If $pattern_{pk}(M, T) = pattern_{pk}(N, T)$, then there exists an $[\![M]\!]_\eta \cong_{\mathtt{O}_x^{M,T}} [\![N]\!]_\eta$.*

PROOF. By assumption and Theorem 15, we know that

$$[\![M]\!]_\eta \cong_{\mathsf{O}_x^{M,T}} \left[\!\!\left[\mathit{pattern}_{pk}\left(M,T\right) \right]\!\!\right]_\eta = \left[\!\!\left[\mathit{pattern}_{pk}\left(N,T\right) \right]\!\!\right]_\eta \cong_{\mathsf{O}_x^{N,T}} [\![N]\!]_\eta$$

Since $M|_{\mathcal{K}_{Pub}} = N|_{\mathcal{K}_{Pub}}$ and $\mathit{pattern}_{pk}\left(M,T\right) = \mathit{pattern}_{pk}\left(N,T\right)$, the oracle $\mathsf{O}_x^{M,T}$ is the same as the oracle $\mathsf{O}_x^{N,T}$. The $\cong_{\mathsf{O}_x^{M,T}}$ relation is transitive (by hybrid argument), and so the result follows. ■

We end by noting that we do not lose generality in this corollary by requiring that $M|_{\mathcal{K}_{Pub}} = N|_{\mathcal{K}_{Pub}}$. If M and N have the same pattern but have different public keys in their parse trees, then we can simply form M' by pairing with M every key in $M|_{\mathcal{K}_{Pub}} \cup N|_{\mathcal{K}_{Pub}}$, and similarly for N'. Since we add only public keys, $\mathit{pattern}_{pk}\left(M',T\right) = \mathit{pattern}_{pk}\left(N',T\right)$. However, it is now the case that $M'|_{\mathcal{K}_{Pub}} = N'|_{\mathcal{K}_{Pub}}$ and the corollary holds.

6 Ideal Encryption

Theorem 18 *Suppose that* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is a computational public-key encryption scheme that provides Dolev-Yao public-key indistinguishability. Then* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *provides Dolev-Yao weak non-malleability.*

PROOF.

Suppose that the theorem is false. Then there is an adversary that is able to produce a message outside the closure of its input set:

$$\begin{aligned}
&\exists\, \mathit{PPT} \text{ adversaries } \mathtt{A},\ \exists \text{ acyclic finite } S \subseteq \mathcal{A},\ \exists M \notin C[S]\,,\\
&\exists \,\text{polynomials } q,\ \text{for infinitely many } \eta:\\
&\Pr[\ t \leftarrow \{0,1\}^\omega\\
&\qquad s \leftarrow [\![S]\!]_\eta^t\,;\\
&\qquad m \leftarrow \mathtt{A}^{\mathtt{M}_\eta^t(\cdot),\mathtt{PbK}_\eta^t(\cdot),\mathtt{PrK}_\eta^t(\cdot),\mathtt{R}_\eta^t(\cdot)}(1^\eta, s):\\
&\qquad m \in \mathit{supp}\ [\![M]\!]_\eta^t \qquad\qquad\qquad] \geq \tfrac{1}{q(\eta)}
\end{aligned}$$

We will construct from this adversary a new adversary $\mathtt{A}_1$ that serves as a counter-example to Theorem 15. But first, consider the parse tree of M. Suppose that every path from the root of the parse tree to a leaf passes through an element of $C[S]$. Then it must be that the root message, M, is in $C[S]$ —

a contradiction. Hence, there must be some path in the parse tree of M such that no element along that path is in $C[S]$, including the leaf M_l.

Now, consider the simple, intermediate adversary $\mathtt{A}_2$, which operates as follows:

(1) It first chooses a random tape $t \leftarrow \{0,1\}^{\omega}$.
(2) It then uses that tape to sample $s \leftarrow [\![S]\!]_{\eta}^{t}$.
(3) It simulates the counter-example adversary $\mathtt{A}$ on input $(1^{\eta}, s)$.
(4) When $\mathtt{A}$ makes an oracle query, $\mathtt{A}_2$ responds appropriately. (Because it knows the random tape t, it can compute any atomic value it wishes, including those returned by the oracles.)
(5) When $\mathtt{A}$ responds with $m \in \mathit{supp}\ [\![M]\!]_{\eta}^{t}$, $\mathtt{A}_2$ uses this to produce a value $m_l \in [\![M_l]\!]_{\eta}^{t}$. That is, it progresses down the path in the parse tree of M that leads to M_l:
   - It starts with a value for $[\![M]\!]_{\eta}^{t}$, and at the root of the parse tree.
   - If the current node is a pair, $M'\,N'$, then it separates the current bit-string value into $[\![M']\!]_{\eta}^{t}$ and $[\![N']\!]_{\eta}^{t}$. It progresses down the path in the parse tree toward M_l, and keeps the value for the new node as its new current value.
   - If the current node is an encryption, $\{\!|M'|\!\}_K$, it uses the tape t to find the value for $[\![K^{-1}]\!]_{\eta}^{t}$. It then uses that to decrypt the current bit-string value to get $[\![M]\!]_{\eta}^{t}$, and progresses down the path in the parse tree toward M_l. (Note: we know that M_l cannot be K, since $K \in C[S]$ and we know this to not be the case for M_l.)

At the end, this adversary will have a value for $[\![M_l]\!]_{\eta}^{t}$. Now, consider what M_l might be:

- M_l cannot be a compound term, since it is a leaf of the parse tree.
- Suppose $M_l \in \mathcal{I}$. Then $M_l \in C[S]$, no matter what S is—a contradiction.
- Suppose $M_l \in \mathcal{K}_{Pub}$. Then, as mentioned above, $M_l \in C[S]$ always.
- Suppose $M_l \in \mathcal{R}$. If $M_l \in \mathcal{R}_{Adv}$ then $M_l \in C[S]$. So, we only need to worry about $M_l \in \mathcal{R} \setminus \mathcal{R}_{Adv}$. There are two cases: either M_l is in the parse tree of something in S, or it is not. The second case leads to a contradiction. If M_l is not in the parse tree of any element of S, then the input to the adversary is completely independent of the required output. Thus, the adversary in question is able to guess a η-bit random value based only on independent input. The probability of this must be bounded above by $2^{-\eta}$, contradicting our assumption that the probability of creating an element of $\mathit{supp}\ [\![M]\!]_{\eta}^{t}$ (and hence an element of $\mathit{supp}\ [\![M_l]\!]_{\eta}^{t}$) is non-negligible.
- Suppose $M_l \in \mathcal{K}_{Priv}$. Then, we proceed similar to above. If $M_l \in \mathcal{K}_{Adv}$, then $M_l \in C[S]$. If $M_l \in \mathcal{K}_{Priv} \setminus \mathcal{K}_{Adv}$ but not in the parse tree of some element of S, the adversary is able to guess a private key based on the corresponding public key, encryptions using the public key, and values independent of the private key. Since we are assuming that the encryption scheme provides

indistinguishability against chosen-ciphertext attacks, the probability of this must be negligible. Again, we find a contradiction.

Thus, the only possibility is that M_l is in $\mathcal{R} \setminus \mathcal{R}_{Adv}$ or in $\mathcal{K}_{Priv} \setminus \mathcal{K}_{Adv}$, and that M_l is in the parse tree of some element of S. However, it cannot be the case that M_l itself is in S, or that M_l can be produced from S only by separating pairs. (If either of those were true, then M_l would be in $C[S]$ itself, a contradiction.) Thus, M_l must only appear in S in the plaintext of encryptions.

Thus, we have an adversary $\mathtt{A}_2$ which takes an element of $[\![S]\!]^t_\eta$ and produces the plaintext to some encryption in S. Granted, $\mathtt{A}_2$ created $[\![S]\!]^t_\eta$ itself and knows every secret. Hence $\mathtt{A}_2$ does not serve as the counter-example to anything. However, a simple modification to $\mathtt{A}_2$ will serve as a counterexample to Theorem 15. Let:

$$S' = \begin{cases} S \cup \{M_l\} & \text{if } M_l \in \mathcal{R} \\ S \cup \left\{N_p, \{\!|N_p|\!\}_{M_l^{-1}}\right\} & \text{(where } N_p \in \mathcal{R}_{Adv}\text{) if } M_l \in \mathcal{K}_{Priv} \end{cases}$$

Then we will be able to distinguish between $[\![S']\!]_\eta$ and $\left[\!\left[pattern_{pk}\left(S', T\right)\right]\!\right]_\eta$ where T is $M|_{\mathcal{K}_{Pub}} \setminus S|_{\mathcal{K}_{Pub}}$. (Note that if M_l is a private key, then it is in neither $C[S]$ or T. Hence the encryption $\{\!|N_p|\!\}_{M_l^{-1}}$ will become $\langle\!|\mathcal{R}|\!\rangle_{M_l^{-1}}$ in $pattern_{pk}\left(M, T\right)$.)

Consider the adversary $\mathtt{A}_1$ that does the following:

(1) It receives as input the value d, which is drawn either from $[\![S']\!]^t_\eta$ or from $\left[\!\left[pattern_{pk}\left(S', T\right)\right]\!\right]^t_\eta$ (for some tape t). It separates d into d_S and d_{test}, where $d_S \in [\![S]\!]^t_\eta$ and either $d_{M_l} \in [\![M_l]\!]^t_\eta$ if M_l is a nonce, or $d_{M_l} \in \left[\!\left[\left\{N_p, \{\!|N_p|\!\}_{M_l^{-1}}\right\}\right]\!\right]^t_\eta$ or $\left[\!\left[\left\{N_p, \langle\!|\mathcal{R}|\!\rangle_{M_l^{-1}}\right\}\right]\!\right]^t_\eta$ if M_l is a private key.

(2) It simulates $\mathtt{A}$ on $(1^\eta, d_S)$. (We will postpone consideration of any oracle calls that $\mathtt{A}$ makes for one moment.)

(3) When $\mathtt{A}$ returns m, $\mathtt{A}_1$ will attempt to extract the value $[\![M_l]\!]^t_\eta$ from m. That is, it recurses down the parse tree of M to M_l, separating pairs and decrypting encryptions, until it arrives at M_l:
   - If $M = N_1\, N_2$ and $m = \langle n_1, n_2, \text{“pair”}\rangle$, then $\mathtt{A}$ continues recursively on n_1 or n_2 depending on whether N_1 or N_2 is on the path to M_l.
   - If $M = \{\!|N|\!\}_K$, $m = \langle c, k, \text{“enc”}\rangle$ and $k \in [\![K]\!]^t_\eta$, then $\mathtt{A}_1$ sends (c, k) to the decryption oracle. Will the decryption oracle decrypt? There are two cases:
     - By definition, $K \in M|_{\mathcal{K}_{Pub}}$. If $K \notin S|_{\mathcal{K}_{Pub}}$ also, then $K \in T$. Hence, the oracle of Defintion 15 will decrypt c.
     - If $K \in S|_{\mathcal{K}_{Pub}}$, then $K \in S|_{\mathcal{K}_{Pub}} \setminus (M|_{\mathcal{K}_{Pub}} \setminus S|_{\mathcal{K}_{Pub}})$. But $S|_{\mathcal{K}_{Pub}} \setminus (M|_{\mathcal{K}_{Pub}} \setminus S|_{\mathcal{K}_{Pub}}) = S|_{\mathcal{K}_{Pub}} \setminus T$. Hence, the decryption oracle of

Definition 15 will decrypt c if c is not in $vis_{[T]^t_\eta}(d)$. However, could c be visible in d with respect to $[\![T]\!]^t_\eta$? If it is, then by the definition of visibility, $\{\!|N|\!\}_K \sqsubseteq pattern_{pk}(S,T)$. In this case, however, $T = M|_{\mathcal{K}_{Pub}} \setminus S|_{\mathcal{K}_{Pub}}$, and so contains no keys in the parse tree of S. Allowing the adversary to decrypt with respect to T does not give it more information about S. Hence, $pattern_{pk}(S,T) = pattern_{pk}(S)$. Thus, if $\{\!|N|\!\}_K \sqsubseteq pattern_{pk}(S,T)$ then $\{\!|N|\!\}_K \sqsubseteq pattern_{pk}(S)$ and so by Theorem 9 it must be that $\{\!|N|\!\}_K \in C[S]$. However, this contradicts the assumption that no node on the path from M to M_l is in $C[S]$, and so c cannot be visible in d. Hence, the decryption oracle of Definition 15 will decrypt it.

Thus, the decryption oracle will always return p, the plaintext of c. $\mathtt{A}_1$ then moves down the parse tree to the node for N and recursively applies this process to p.

(4) If any of the above conditions fail, then $\mathtt{A}$ immediately stops and outputs 0. Otherwise, $\mathtt{A}_1$ will acquire a value m_l which may be the encoding of M_l. $\mathtt{A}_1$ tests this using the string d_{test}, which it reserved at the beginning. If M_l is a nonce, then d_{test} will be the value for M_l; $\mathtt{A}$ can simply test that $m_l = d_{test}$. If M_l is a private key, then d_{test} contains a plaintext $[\![N_p]\!]^t_\eta$ and an encryption of that plaintext. $\mathtt{A}_1$ simply decrypts the encryption with m_l. If the result should be the same as the other value of d_{test}. If these tests are satisfied, then $\mathtt{A}_1$ outputs 1. Otherwise, it outputs 0.

$\mathtt{A}_1$ will return 1 whenever $\mathtt{A}$ produces an element of $supp\ [\![M]\!]^t_\eta$. Hence, $\mathtt{A}_1$ will return 1 with probability at least $\frac{1}{q(\eta)}$ given that d is in fact drawn from $[\![S]\!]_\eta$. If, on the other hand, d is drawn from $\left[\!\left[pattern_{pk}(M,T)\right]\!\right]_\eta$, then $\mathtt{A}$ cannot have a non-negligible chance of producing a $m \in supp\ [\![M]\!]^t_\eta$. Since $M_l \notin C[S]$, it cannot be that $M_l \sqsubseteq pattern_{pk}(S) = pattern_{pk}(S,T)$.

- If M_l is a nonce, then this implies that the sample d will be entirely independent of the actual value for $[\![M_l]\!]^t_\eta$.
- If M_l is a private key, on the other hand, then d may include encryptions made using the public key $\left[\!\left[M_l^{-1}\right]\!\right]^t_\eta$. But the encryption provides indistinguishability against chosen-ciphertext attack, so it is infeasible to recover a private key using only encryptions under the corresponding public key. Since d is otherwise independent of $[\![M_l]\!]^t_\eta$, $\mathtt{A}$ cannot have a non-negligible chance of recovering $[\![M_l]\!]^t_\eta$.

Thus, the probability that $\mathtt{A}_1$ will return 1 given that d is sampled from $\left[\!\left[pattern_{pk}(S,T)\right]\!\right]^t_\eta$ must be negligible.

Hence, if $\mathtt{A}$ has a non-negligible chance of constructing $m \in supp\ [\![M]\!]^t_\eta$ from a sample from $[\![S]\!]^t_\eta$, then $\mathtt{A}_1$ has a non-negligible chance of distinguishing $[\![S']\!]_\eta$

from $\left[pattern_{pk}\left(S', T\right)\right]_{\eta}$, a contradiction of Theorem 15.

There remains only one last complication: $\mathtt{A}$ has access to oracles while operating. In particular, $\mathtt{A}$ can request any public key, any private key in $\mathcal{K}_{Adv}$, any identifier, and any nonce in $\mathcal{R}_{Adv}$. How does $\mathtt{A}_1$ respond to these oracle calls when it simulates $\mathtt{A}$?

The answer is that we slightly modify the set S' to include the information needed to respond. In particular, let $S|_{\mathcal{K}_{Pub}}$ and $S|_{\mathcal{R}_{Adv}}$ be defined analogously to $S|_{\mathcal{K}_{Pub}}$.Then the set S' is will actually be

$$S' = \begin{cases} S \cup S|_{\mathcal{K}_{Pub}} \cup S|_{\mathcal{K}_{Adv}} \cup S|_{\mathcal{R}_{Adv}} \cup \{M_l\} & \text{if } M_l \in \mathcal{R} \\ S \cup S|_{\mathcal{K}_{Pub}} \cup S|_{\mathcal{K}_{Adv}} \cup S|_{\mathcal{R}_{Adv}} \cup \left\{N_p, \{\!|N_p|\!\}_{M_l^{-1}}\right\} & \text{if } M_l \in \mathcal{K}_{Priv} \end{cases}$$

When $\mathtt{A}_1$ receives the input d it strips off d_S as before and simulates $\mathtt{A}$. When $\mathtt{A}$ makes an oracle call, however, $\mathtt{A}_1$ can respond:

- If the oracle is being asked for an identifier, $\mathtt{A}_1$ computes the representation of that identifier. (As mentioned before, we assume that the encoding of identifiers is efficiently computable.)
- If the oracle is being called on an ingredient of S, then the additional information in s contains the needed bit-string.
- Otherwise, the needed value is a random variable independent of d. $\mathtt{A}_1$ can sample from the relevant distribution to produce an indistinguishable value. It then stores the value for future use (and if the value is a key, the corresponding secret or public key also), and returns it.

Since the formal messages we added to S' are already in $C[S]$, they do not change the pattern of the original S. Hence, adding them to S' does not change the distribution of d_S, and $\mathtt{A}$ will progress as before. ■

7 Conclusion and Open Problems

The primary contribution of this paper is three-fold:

(1) First, we presented a definition of Dolev-Yao weak non-malleability, which directly captures the main assumptions of the Dolev-Yao model.
(2) We then translated the definition of Dolev-Yao indistinguishability from the secret-key to the public-key setting, and showed that it is satisfied by encryption that provides indistinguishability under the chosen-ciphertext attack.
(3) Lastly, we showed that Dolev-Yao indistinguishability implies Dolev-Yao weak non-malleability.

One obvious extension of this work would be to examine the relationship between Dolev-Yao indistinguishability and non-malleability further. In many settings, non-malleability is either equivalent to or strictly stronger than indistinguishability. The fact that Dolev-Yao indistinguishability implies Dolev-Yao non-malleability is strong evidence for their equivalence in this setting, but the question remains open.

Another interesting way to extend this work would be to strengthen the definition of Dolev-Yao non-malleability. The current definition states, informally, that the adversary has only a negligible chance of "hitting" a given target (i. e., producing an encoding of a given M). If possible, it would be interesting to find an encryption scheme that keeps the adversary from hitting *any* target:

Definition 19 (Strong Dolev-Yao non-malleability) *A computational encryption scheme provides* strong Dolev-Yao non-malleability *if, when used in the* $[\![\cdot]\!]_\eta^t$ *operation, the adversary cannot create anything outside the closure:*

$$
\begin{aligned}
&\forall\, \textit{PPT adversaries } \mathtt{A},\ \forall\, \textit{finite, acyclic } S \subseteq \mathcal{A}, \\
&\forall\, \textit{polynomials } q,\ \forall\ \textit{sufficiently large } \eta : \\
&\Pr[\ t \leftarrow \{0,1\}^\omega \\
&\qquad s \leftarrow [\![S]\!]_\eta^t\,; \\
&\qquad m \leftarrow \mathtt{A}^{\mathcal{M}_\eta^t(\cdot), \mathcal{PbK}_\eta^t(\cdot), \mathcal{PrK}_\eta^t(\cdot), \mathcal{R}_\eta^t(\cdot)}(1^\eta, s) : \\
&\qquad \exists M \in \mathcal{A} \setminus C[S]\,.m \in \textit{supp } [\![M]\!]_\eta^t \qquad] \leq \tfrac{1}{q(\eta)}
\end{aligned}
$$

A third way to improve this work would be to remove the requirment that S be acyclic. (This would most likely also remove the same assumption from the results of Abadi and Rogaway [1].)

Lastly, it would interesting to incorporate into this approach cryptographic operations other than encryption, such as hashes and signatures.

8 Acknowledgments

A previous, and weaker, form of this paper appeared as part of [9]. I thank my co-authors, Silvio Micali and Moses Liskov, for their permission to publish this extension and for many helpful discussions. Also, I would like to thank Joshua Guttman and Amy Herzog for their unflagging support.

References

[1] 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, *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer-Verlag, August 2000.

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

[3] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proceedings, 10th ACM conference on computer and communications security (CCS)*, October 2003. Full version available at `http://eprint.iacr.org/2003/015/`.

[4] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Krawczyk [11], pages 26–45. Full version found at `http://www.cs.ucsd.edu/users/mihir/papers/relations.html`.

[5] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In D. Stinson, editor, *Advances in Cryptology - CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, August 1993. Full version of paper available at `http://www-cse.ucsd.edu/users/mihir/`.

[6] R Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [11], pages 13–25.

[7] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[8] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. Workshop on Formal Methods and Security Protocols (FMSP'99), July 1999.

[9] Jonathan Herzog, Moses Liskov, and Silvio Micali. Plaintext awareness via key registration. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–564. Springer-Verlag, August 2003.

[10] Naoki Kobayashi and Benjamin C. Pierce, editors. *Formal Eavesdropping and Its Computational Interpretation*, volume 2215 of *Lecture Notes in Computer Science*. Springer, 2001.

[11] H. Krawczyk, editor. *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, August 1998.

[12] P. D. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the 5th ACM Conference on Computer and Communication Security (CCS '98)*, pages 112–121, November 1998.

[13] P. D. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security protocols. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *World Congress on Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 776–793. Springer, September 1999.

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

[15] P. Mateus, J.C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In Roberto M. Amadio and Denis Lugiez, editors, *Proceedings, 14th International Conference on Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 323–345. Springer, 2003.

[16] Catherine Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communication*, 21(1):44–54, January 2003.

[17] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. Workshop on Issues in the Theory of Security (WITS '02), January 2002.

[18] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

[19] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings, Theory of Cryptography Conference*, number 2951 in Lecture Notes in Computer Science, pages 133–151. Springer, February 2004.

[20] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report). In *Proc. 17th Annual Conference on the Mathematical Foundations of Programming Semantics (MFPS 2001)*, volume 45 of *Electronic Notes in Theoretical Computer Science*, May 2001.

[21] J. C. Mitchell, M. Mitchell, and A. Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *39th Annual Syposium on Foundations of Computer Science (FOCS 1998)*, pages 725–733. IEEE Computer Society, November 1998.

[22] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 427–437, May 1990.

[23] L C Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[24] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *Proceedings, Foundations of Software Technology and Theoretical Computer Science (FST TCS 2003)*, volume 2914 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2003.

[25] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Syposium on Foundations of Computer Science (FOCS 1999)*, pages 543–553. IEEE Computer Society, October 1999.

[26] D. Song. Athena, an automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW 12)*, pages 192–202. IEEE Computer Society, June 1999.