A Computational Interpretation of Dolev-Yao Adversaries

Jonathan Herzog

Massachusetts Institute of Technology
jherzog@mit.edu

Abstract. The Dolev–Yao model is a simple and useful framework in which to analyze security protocols, but it assumes an extremely limited adversary. It is unclear if the results of this model would remain valid were the adversary to be given additional power. In this work, 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 two steps:

1. We translate the allowed behaviors of the Dolev-Yao adversary into the computational model, an alternate framework with a very powerful adversary.
2. We show that sufficiently strong computational cryptography can limit the computational adversary to these behaviors.

1 Introduction

How can we tell if a cryptographic protocol is correct? That is, 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 the protocol are assumed to be bit-strings from some distribution, and the adversary is assumed to be an arbitrary algorithm. The cryptography is assumed to be an algorithm (or tuple of algorithms) which satisfy some asymptotic property even in the presence of an arbitrary adversary.

To prove a protocol correct in this model, one would show a *reduction* from the protocol to the underlying cryptography. 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 the underlying cryptography. Hence, one can conclude that if the cryptography is secure then the protocol must be secure also. (See [3] 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

[1] That is, it has access to a random coin and executes in expected time polynomial in the length of its input. The notation $\mathtt{A}_{PPT}$ denotes that algorithm $\mathtt{A}$ runs in probabilistic polynomial time.

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 manually produced for each protocol.

Fortunately, there are alternate models — one of them being the *Dolev-Yao* model [6]. In this setting, messages are assumed to be elements of some abstract algebra, and cryptography 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. Since all participants (honest and malicious) can be represented as state machines, they can be composed together to produce a single large "system" machine. Security properties can be expressed as safety properties about this machine, and such properties can be verified automatically. Although the general problem of protocol security is undecidable [7], several automated tools have been successfully used. (See [10, 12, 13] for typical examples.)

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 operations from which it can choose is fixed and quite small. It is unclear whether security against this restricted adversary implies security against more realistic adversary models.

Hence, one must seemingly choose between the simplicity of the Dolev-Yao model and the meaningfulness 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 shall only discuss one small part: the adversary. In particular, we will show that sufficiently strong computational cryptography forces an equivalence of sorts between the Dolev-Yao adversary and all computational adversaries.

This question has already been partially addressed by Abadi and Rogaway [1], who showed that the concept of *indistinguishability* in the Dolev-Yao model can be meaningful in the computational one. That is, they showed that if two Dolev-Yao messages are indistinguishable to the Dolev-Yao adversary, then they can be translated to two computational messages which will be indistinguishable to the computational one.

We continue and build upon this work by considering the issue 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. Although this does not give the Dolev-Yao model a complete grounding in computational cryptography (for reasons discussed in Section 5) it represents a significant step in that direction.

2 The Dolev–Yao Model

We begin by exploring the powers of the adversary in the Dolev-Yao model.

In this setting, messages are assumed to be elements of an algebra $\mathcal{A}$ of values. There are two types of atomic messages:

– Texts ($\mathcal{T}$) with two subtypes: names (public, predictable, denoted by $\mathcal{M}$) and nonces (private, random, unpredictable, denoted by $\mathcal{R}$), and
– Keys ($\mathcal{K}$) with two subtypes: public keys ($\mathcal{K}_{Pub}$) and private keys ($\mathcal{K}_{Priv}$)

Compound messages are created by two deterministic operations:

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

Additionally, the algebra is assumed to be *free*: every value has a unique representation.

In this 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. The model contains only the messages they send and receive; internal states are not modeled explicitly.

The network is assumed to be completely under the control of the adversary, who can record, delete, replay, reroute, and reorder 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. Since we are not interested with the honest participants, we abstract them into an "environment" process. 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 ($r_i \subseteq \mathcal{A}$):

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

We will say nothing here about how the environment produces the responses. We will say a great deal, on the other hand, about how the adversary produces the queries.

Intuitively, any query q_i must be *derivable* from what is known initially and $r_0, r_1, r_2 \ldots r_{i-1}$. The initial knowledge set of the adversary includes at least the following:

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

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 to S 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 "join" element into its component elements.

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. $\mathcal{M} \cup \mathcal{K}_{Pub} \cup \mathcal{K}_{Subv} \cup \mathcal{R}_{Adv} \subseteq C[S]$,
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). *If the Dolev-Yao adversary knows a set* S *of messages, it can only produce messages in* $C[S]$.

Hence, in the Dolev-Yao model, it must be that $q_i \in C[S_0 \cup r_0 \cup \ldots \cup r_{i-1}]$ where S_0 are the messages the adversary knows at the beginning of the execution. The main result of this paper is that computational cryptography can limit the computational adversary to this closure operation.

3 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 message outside the closure of its input. That is, a computational adversary $\mathtt{A}_{PPT}$ should have only a small chance of transforming a set of messages S into a message outside of $C[S]$:

Attempt 1. *An abstract encryption operator is* ideal *if* $\forall \mathtt{A}_{PPT}$, $\forall S \subseteq \mathcal{A}$,

$$\Pr[M \leftarrow \mathtt{A}(S) : M \in (\mathcal{A} \setminus C[S])] \textit{ is small.}$$

(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, what is meant is 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 with it:

– It is unclear how a set S of Dolev-Yao messages can be passed as input to a computational adversary, for example, 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 computational "messages:" probability distributions on bit-strings. While the material of this section is not exactly that of Abadi and Rogaway [1], it is an exact analogue of their work — adapted from the case of symmetric encryption to that of public-key encryption.

The "encoding" of a message M, written $[\![M]\!]_\eta^t$, depends on a random tape[2], a security parameter, and an arbitrary public-key encryption scheme[3]:

Definition 3 (Encoding). *Let $\eta \in \mathsf{N}$ 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 $M \in \mathcal{A}$, $[\![M]\!]_\eta^t$ is defined recursively as:*

- *If $M \in \mathcal{R}$ is a nonce, then $[\![M]\!]_\eta^t = \langle \sigma_M, \text{"nonce"} \rangle$, where σ_M is the value of the tape partition associated with M.*
- *If (M, M^{-1}) is a public/private key pair, then $[\![M]\!]_\eta^t = \langle e, \text{"pubkey"} \rangle$ and $[\![M]\!]_\eta^t = \langle d, \text{"privkey"} \rangle$ where (e, d) is the output of $\mathsf{G}(\eta, \sigma_M)$. Note that σ_M is used for randomness.*
- *If $M \in \mathcal{M}$ is a name, then $[\![M]\!]_\eta^t$ is mapped to $\langle m, \text{"name"} \rangle$ where m is any (short) bit-string uniquely associated with M. That is, we do not care how names are mapped to bit-strings so long as each name is uniquely represented.*
- *If $M = pair(M_1, M_2)$, then $[\![M]\!]_\eta^t$ is the mapping from pairs of distributions to distributions given by $\left\langle [\![M_1]\!]_\eta^t, [\![M_2]\!]_\eta^t, \text{"pair"} \right\rangle$.*
- *If $M = encrypt(M', K)$ is an encryption, then $[\![M]\!]_\eta^t$ is the mapping from pairs of distributions to distributions given by $\left\langle \mathsf{E}\left([\![M']\!]_\eta^t, \rho, [\![K]\!]_\eta^t\right), \text{"enc"} \right\rangle$ where $\rho \leftarrow \mathsf{Randomness}$*

That is, 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.[4] Compound terms are made via either bit-string concatenation or a computational encryption scheme. Note that the coin flips used by the encryption

[2] An element of $\{0,1\}^\omega$, infinite sequences of bits.

[3] In the computational setting, public-key encryption is not a simple operation but a triple of algorithms:

- $\mathsf{G} : \mathsf{Parameter} \times \mathsf{Randomness} \to \mathsf{PublicKey} \times \mathsf{PrivateKey}$ is the key generation algorithm,
- $\mathsf{E} : \mathsf{String} \times \mathsf{Randomness} \times \mathsf{PublicKey} \to \mathsf{Ciphertext}$ is the encryption algorithm, and
- $\mathsf{D} : \mathsf{String} \times \mathsf{PrivateKey} \to \mathsf{String} \cup \{\perp\}$ is the decryption algorithm.

Note that the key generation and encryption algorithms are randomized.

[4] It is possible to strengthen the adversary by allowing it to pick the values of its own nonces and keys. This technical issue contains no interesting details, and is discussed in [9].

algorithm are *not* taken from the tape. Hence, $[\![encrypt(M', K)]\!]_\eta^t$ remains a distribution even if t is fixed.

Now that we have introduced the security parameter, we can refine our notion of "small" probabilities. By this, we mean *negligible*:

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

Hence, we can now re-attempt to translate Definition 2 into computational terms:

Attempt 2. *An encryption scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* ideal *if, when used in* $[\![\cdot]\!]_\eta^t$,

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

(Here, $\mathbf{supp}D$ means the support of distribution D.)

However, this definition is still problematic, for two technical reasons. First, there may be an infinite number of elements in $\mathcal{M}$, $\mathcal{R}_{Adv}$, $\mathcal{K}_{Pub}$ and $\mathcal{K}_{Subv}$. We represent access to an infinite amount of information via oracles:

- $\mathtt{M}_t(x)$ returns (the encoding of) the name of the xth participant.
- $\mathtt{PbK}_t(\cdot)$ returns the public key of any principal,
- $\mathtt{PrK}_t(\cdot)$ returns the private key of any subverted principal, and
- $\mathtt{R}_t(\cdot)$ returns the (encoding of) any nonce in $\mathcal{R}_{Adv}$.

Second, our results require a technical limitation: that S be acyclic. A set of messages 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. (A rigorous definition can be found in [1].) Our results build on those of Abadi and Rogaway [1], and we require acyclic sets simply because they do. However, as discussed in [9], this is a realistic assumption for the traffic of most "real-world" protocols.

Hence, we arrive at our final[5] security condition:

Definition 5 (Ideal Encryption). *An encryption scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* ideal *if, when used in* $[\![\cdot]\!]_\eta^t$,

$$\begin{array}{l}
\forall\, \mathtt{A}_{PPT},\ \forall\, \textit{acyclic } S \subseteq \mathcal{A},\ \forall M \notin C[S], \forall\, \textit{polynomials } q,\ \forall\ \textit{sufficiently large } \eta : \\
\Pr[\ t \leftarrow \{0,1\}^\omega \\
\quad s \leftarrow [\![S]\!]_\eta^t\ ; \\
\quad m \leftarrow \mathtt{A}^{M_t(\cdot), PbK_t(\cdot), PrK_t(\cdot), R_t(\cdot)}(1^\eta, s) : \\
\quad m \in \textit{\textbf{supp}}\,[\![M]\!]_\eta^t \qquad\qquad\qquad\qquad] \leq \frac{1}{q(\eta)}
\end{array}$$

[5] In [9], we consider the further complication where the adversary is not required to produce something actually in $\mathbf{supp}\,[\![M]\!]_\eta^t$ but merely something that *appears* to be in it to an honest participant. We omit consideration of this uninteresting complication here.

In the next section, we show that there exist ideal encryption schemes in a standard computational model.

4 Ideal Encryption

To show that ideal cryptography can be instantiated, we use a standard computational definition of security known as *plaintext awareness* [4, 2]. This requires the presence of an extra party:

Definition 6 (Random Oracle). *The* random oracle *is an oracle, accessible by all parties, which provides a random mapping from all possible inputs to bit-strings of some fixed length.*

We will not formally define plaintext awareness here, but use only the underlying intuition:

Definition 7 (Plaintext Awareness (informal)). *Let* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *be an encryption scheme where both the* E *and* D *are allowed to make private queries to the random oracle. Then the scheme is* plaintext aware *if*

1. *It provides confidentiality (i.e.,* semantic security *[8]), and*
2. *There exists an algorithm* $\mathtt{X}$*, called the* extractor*, that can (almost always) determine the plaintext of any ciphertext produced by any adversary given only the ciphertext itself, the public key purportedly used to encrypt, and the queries made to the random oracle by the adversary.*

Plaintext awareness implies that every adversary must be "aware" of the plaintext to every ciphertext it creates. (Suppose an adversary creates the encryption of an unknown plaintext. Then it can always determine the plaintext simply by running the extractor itself.) Note that without the random oracle, plaintext awareness cannot exist: extracting the plaintext without the oracle queries is the same as simply decrypting a ciphertext. If one assumes the presence of a random oracle, however, plaintext awareness can actually be achieved and is the strongest known definition of security for asymmetric encryption.

Theorem 1. *Suppose that* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is plaintext-aware. Then* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is ideal.*

Proof Sketch. We provide a full proof in [9] and merely sketch the proof here: Suppose there is an adversary that violates the condition. That is,

$$\begin{array}{l}\exists\, \mathtt{A}_{PPT},\ \exists \text{acyclic } S \subset \mathcal{A},\ \exists\, M \notin C\,[S]\,,\ \exists \text{a polynomial } q,\ \exists \text{ infinitely many } \eta \\ \Pr[\ t \leftarrow \{0,1\}^{\omega} \\ \qquad s \leftarrow [\![S]\!]_{\eta}^{t}\,; \\ \qquad m \leftarrow \mathtt{A}^{\mathtt{PbK}_t(\cdot),\mathtt{PrK}_t(\cdot),\mathtt{M}_t(\cdot),\mathtt{R}_t(\cdot)}(1^{\eta}, s)\ : \\ \qquad m \in \mathbf{supp}\,[\![M]\!]_{\eta}^{t} \qquad\qquad\qquad\qquad] \geq \frac{1}{q(\eta)}\end{array}$$

Consider the parse tree of M. The root of this tree is M, which is not in $C\,[S]$. It may be, however, that some nodes of M's parse tree *are* in $C\,[S]$. Suppose, however, that every path from root to leaf contains some node in $C[S]$. Recall that $C[S]$ is closed under pairing and under encryption with keys in $C[S]$. Hence membership in $C[S]$ is closed up the parse tree: if every path from root to leaf contained a node in $C[S]$, the root itself must be in $C[S]$. But we know that the root is M itself, which is not in $C[S]$.

So the parse tree of M contains some path from root to leaf which contains no element in $C[S]$. Along this path, it must be the case that if the adversary can create an encoding of an internal node, then it can create encodings of both children:

- If the interior node is a pair, then our encoding makes it easy to separate the encoding of the pair into the encodings of the components.
- If the interior node is a Dolev-Yao encryption, on the other hand, and the adversary can create an encoding, then the adversary creates a computational ciphertext. Plaintext awareness tells us that the adversary can also produce the corresponding plaintext, which will also be an encoding of the Dolev-Yao plaintext. Since all public keys are known to the adversary, it can produce the other child of this interior node also.

Thus, if the adversary can produce an encoding of M with some probability, it must also be able to produce, with almost the same probability, the encoding of some leaf M' of M's parse tree. Since this leaf is at the end of a path lacking any nodes in $C[S]$, the leaf node is outside $C[S]$ also. Furthermore, all leaf nodes are atomic elements of $\mathcal{A}$. For an atomic element to be outside $C[S]$, it cannot be either public information or private information created by the adversary. Hence, M' is either an honest nonce or private key, both of which are random values.

There are two cases:

1. In the first case, M' is somehow related to the input set S. It can be that M' is in the parse tree of something in S, or that M' is a private key and the corresponding public key is in the parse tree of something in S. In either case, the ability to produce the single element of $\mathbf{supp}\, [\![M']\!]_\eta^t$ breaks the confidentiality of the computational encryption. (This reduction is quite involved, and depends crucially on the asymmetric-encryption analogue to the indistinguishability result of Abadi and Rogaway[1].)
2. In the other case, M' is completely unrelated to S, making $[\![M']\!]_\eta^t$ and $[\![S]\!]_\eta^t$ independent distributions. In this case, the adversary is able to guess a nonce or private key from completely independent input. However, we know that the odds of guessing any nonce or private key must be negligible.

Hence, we reach a contradiction in both cases, and the adversary could not have a non-negligible chance of creating something in $\mathbf{supp}\, [\![M]\!]_\eta^t$.

5 Conclusion and Open Problems

The primary contribution of this paper is two-fold:

1. First, a rigorous computational definition of ideal cryptography, a central assumption of the Dolev-Yao model.
2. Second, a proof that this definition can be met using standard definitions of computational cryptography.

One way in which this work can be extended would be to remove the random oracle. The presence of such an oracle is a somewhat controversial assumption in computational cryptography, and removing it from our results would only increase the meaningfulness of the Dolev-Yao model. (Such a result would require that ideal cryptography be instantiated with weaker schemes, such as those only shown to be secure against the chosen-ciphertext attack [2]).

Another way to improve this work would be to remove the assumption that S be acyclic. This would most likely involve removing the same assumption from the results of Abadi and Rogaway [1].

A third interesting way to extend this work would be to strengthen the definition of ideal cryptography. 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 8. *An encryption scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ *is* unversally ideal *if the adversary cannot create anything outside the closure:*

$$\begin{array}{l} \forall\, \mathtt{A}_{PPT},\ \forall\, \textit{acyclic } S \subseteq \mathcal{A}, \forall\, \textit{polynomials } q,\ \forall\ \textit{sufficiently large } \eta : \\ \Pr[\ t \leftarrow \{0,1\}^{\omega} \\ \quad s \leftarrow [\![S]\!]_{\eta}^{t}\ ; \\ \quad m \leftarrow \mathtt{A}^{M_t(\cdot), PbK_t(\cdot), PrK_t(\cdot), R_t(\cdot)}(1^{\eta}, s) : \\ \quad \exists M \notin C\,[S]\,.m \in \boldsymbol{supp}[\![M]\!]_{\eta}^{t} \qquad\qquad] \leq \frac{1}{q(\eta)} \end{array}$$

Lastly, and most importantly, it would be interesting to extend this work to consider the honest participants. This work abstracts the honest participants into an environment which is implicitly assumed to never encounter error conditions. However, real adversaries can always intentionally create "garbage" messages. What should an honest participant do if it encounters one? Immediate answers would include termination, response with an error code, or both. However, all three of these actions are observable to the adversary. That is, they all release at least one bit of information, turning the honest participants into (highly specialized) oracles. There is no guarantee that an ideal encryption scheme will remain secure if the adversary has access to such oracles. It is important to consider the actions that honest participants take when the unexpected occurs; otherwise secure protocols and implementations have been successfully attacked through an exploitation of error conditions [11, 5].

References

1. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference*

on *Theoretical Computer Science (IFIP TCS2000)*, number 1872 in Lecture Notes in Computer Science, pages 3–22. Springer-Verlag, 2000.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology (CRYPTO 98)*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer–Verlag, 1998. Full version found at http://www.cs.ucsd.edu/users/mihir/papers/relations.html.
3. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In D. Stinson, editor, *Advances in Cryptology–Crypto 93 Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer–Verlag, 1993. Full version of paper available at http://www-cse.ucsd.edu/users/mihir/.
4. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption– how to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology – Eurocrypt 94 Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
5. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, 1998.
6. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
7. 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. Available at `http://www.cs.bell-labs.com/who/nch/fmsp99/program.html`.
8. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, (28):270–299, 1984.
9. Jonathan Herzog. Computational soundness for formal adversaries. Master's thesis, Massachusetts Institute of Technology, October 2002.
10. 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.
11. James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding OAEP as standardized in PKCS #1 v2.0. In *Advances in Cryptology — CRYPTO 2001*, number 2139 in Lecture Notes in Computer Science, pages 230–238, 2001.
12. L C Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
13. D. Song. Athena, an automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 192–202, 1999.