INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Modelling an Attacker With Cryptanalytical Capabilities

Bruno José Conchinha Montalto

Dissertação para obtenção do Grau de Mestre em

Matemática Aplicada e Computação

26 de Setembro de 2008

Resumo

A abordagem simbólica à análise de protocolos de segurança, introduzida por Dolev e Yao há cerca de 25 anos, tem sido muito bem aceite entre a comunidade científica. Estes modelos usam tipicamente a assumpção de "criptografia perfeita", abstraindo os detalhes das primitivas criptográficas utilizadas no protocolo. Esta simplicidade permitiu o desenvolvimento de várias ferramentas automáticas de análise da segurança de protocolos baseadas nesta abordagem. Contudo, é difícil justificar que estas abstracções são correctas, pois na prática as primitivas criptográficas têm propriedades que podem ser exploradas pelo atacante.

A abordagem computacional da análise de protocolos de segurança resolve este problema, tratando as primitivas criptográficas como algoritmos e utilizando conceitos como os de complexidade e probabilidade. Tais modelos são, contudo, bastante complexos, e é geralmente difícil provar teoremas neste contexto.

O objectivo deste trabalho é desenvolver uma ferramenta simbólica para a análise de protocolos de segurança que permita ao atacante explorar propriedades conhecidas das primitivas criptográficas. Vamos mostrar como representar propriedades criptográficas e informação parcial sobre mensagens secretas. Vamos ainda estudar como obter uma estimativa da probabilidade de sucesso de um ataque e avaliar o impacto de propriedades criptográficas na segurança do protocolo.

Palavras-chave: Protocolo, atacante, mensagem, primitiva criptográfica, propriedade criptográfia, entropia de Shannon

Abstract

The formal approach to the analysis of security protocols, introduced by Dolev and Yao 25 years ago, has been very popular among the scientific community. These models tipically use the "perfect criptography" assumption, abstracting the specific details of the cryptographic primitives used in the protocol. This simplicity has allowed the development of several tools for automated analysis of protocols based on this approach. However, it is hard to justify that these abstractions are sound, since in practice cryptographic primitives have properties which may be explored by the attacker.

Computational approaches deal with this problem, by treating cryptographic primitives as algorithms and dealing with concepts like complexity and probability. Computational models, however, are quite complex, and it is generally hard to prove theorems in such a setting.

The goal of this work is to develop a formal framework for the analysis of security protocols which allows the attacker to use known properties of cryptographic primitives. We will show how to represent cryptographic properties and partial information about secret messages. We will also study how to estimate the probability of success of an attack and assess the impact of cryptographic properties on the security of protocols.

Keywords: Protocol, attacker, message, cryptographic primitive, cryptographic property, Shannon entropy

Acknowledgements

I would like to leave here my thanks to the people whose support and affection have given me the strength and ability to complete this thesis.

First of all I must thank my mother, for the support she has always given me through so many hard times. Her wise counsel has always lead me and still leads me in my life, and her strength inspires me to be a better person.

I also leave my greatest thanks to my advisor, Professor Carlos Caleiro. I thank him for estimulating my creativity and for believing in my work; mostly, however, I acknowledge his wisdom and understanding. A thankful word is also due to ETH Zurich's information security group, namely Professor David Basin, Professor Cas Cremers and Simon Meiers, for giving me the opportunity to present my work in their group seminar and for a lot of helpful input.

I should also thank my friends, who have made the last five years so thoroughly enjoyable and whose diverse and extraordinary personalities have helped shape mine: André Vasconcelos, David Henriques, Luís Alexandre, Luís Sousa, Manuel Martins, Nuno Freitas and Rui Palma.

Finally I must leave a special thanks to Lídia del Rio, for her unending support during the last year and for always finding a new kind of smile to share.

À minha mãe, avós e irmãos

Contents

Chapter 1

Introduction

The analysis of security protocols has been the subject of much research in the last few decades. Two fundamentally distinct approaches have been used to this end.

In the formal approach, introduced in the early 1980's by Andrew Dolev and Danny Yao [10], technical details of cryptographic primitives are ignored. These models adopt the "perfect cryptography" assumption: for example, encryption is viewed as a "black-box" operation, so that, from an encrypted message, the attacker has no way of finding anything about the original message.

These formal methods for analysis of security protocols have been very popular and widely accepted. One of the clearest proofs of the usefulness of such methods is the "man-in-the-middle" attack to the Needham-Schroeder protocol, discovered by Gavin Lowe in 1995 [20] using the Dolev-Yao model. Since the original proposal, many generalizations have been proposed: the range of cryptographic primitives considered has been broadened and some simple algebraic properties of cryptographic functions have been considered. Several tools for automated analysis of security protocols (mostly attack-search engines) have also been developed based on this method and are currently being used in industrial scale projects, although most of them assume a bounded environment [5, 9, 23].

In fact, one of the main reasons for the popularity of this approach is that such strong abstractions allow automated proofs of the security of protocols. The main weakness of formal methods is that it is hard to prove that these abstractions are sound. In fact, realistic cryptographic primitives have properties which the attacker may explore and attack: for an example, encryption functions may reveal the length of the original message. Formal methods usually do not consider this kind of partial information.

In stark contrast with the formal approach is the computational one. Computational methods use a more complex framework, in which messages are treated as actual bitstrings and cryptographic primitives as functions acting on those bitstrings. In order to provide a more complete and realistic representation of a "real" attacker, the computational approach deals with concepts like complexity and probability. [19]

Security proofs in the computational approach are generally stronger than in the formal approach, since they allow the attacker to explore vulnerabilities of the cryptographic primitives to find a suitable attack. However, the greater complexity of these methods makes it

difficult to obtain such proofs in an automated way, and it is hard to analyse even simple protocols in this setting.

In this work we present a formal model in which we allow the attacker to obtain and take advantage of partial information about the messages exchanged in the execution of the protocol by exploring certain cryptographic properties. We develop a framework in which we are able to represent not only protocols but also properties of cryptographic primitives and partial information about a message. The attacker may use such properties to find partial information about secret messages and ultimately a suitable attack. This partial information is represented in the form of probabilistic statements about secret messages seen as random variables. We describe the possible actions for the attacker at each point of the execution of the protocols and use these actions to define attack strategies.

We also present a way to estimate the probability of success of an attack based on the cryptanalytical knowledge which we allow the attacker to use. This is useful not only because it provides a quantitative measure of the security of a protocol, but also because it allows us to evaluate the effect of certain cryptographic weaknesses in the security of that protocol. In fact, our model is based not on the specific details of the cryptographic primitives but rather on their properties known by the attacker and the partial information about secret messages that the attacker may uncover from them. Thus, by describing general cryptographic properties (which may or may not be verified by different cryptographic primitives), we may use our model to assess the impact of those properties on the security of a protocol. Ultimately, the goal (which is not thoroughly achieved in this thesis) will be to use the framework to assess the relative security (or weakness) of protocols against the knowledge available to the attacker about weaknesses of cryptographic primitives.

The presentation is organized as follows. In Section 2 we introduce our framework. We show how to represent in our framework messages, cryptographic primitives and security protocols, and describe the capabilities of the attacker. We assume an active attacker: a legitimate user of the network who may participate in protocol execution, intercept messages and impersonate other agents just as a Dolev-Yao attacker, but may also explore cryptographic properties.

In section 3 we show how we represent cryptographic properties and how we allow the attacker to use them. As an example, we prove a theorem which shows how IND-CPA (indistinguishability under chosen plaintext attacks) security, a standard property for asymmetric encryption schemes, may be translated to our framework. In particular, we will obtain an equivalent but simpler statement of this property. Furthermore, we will show that IND-CPA security has an undesirable property: namely, given an encryption scheme verifying this property, we will show that it is possible to obtain another encryption scheme which is in a strong sense at least as secure as the first one but is no longer IND-CPA secure.

Section 4 concerns the estimation of probabilities based on the cryptographic properties which we allow the attacker to use. Our goal is to find an estimation of the probability distribution of the secret and randomly generated messages (and the relation between them)

that verifies all properties known by the attacker, but does not introduce any additional structure and remains as hard to attack as possible. To achieve this we will use the concept of Shannon entropy[24]. We present some examples which illustrate these ideas and obtain a theorem which may be useful in proving that a certain set of cryptographic properties is not enough to obtain a suitable attack.

In the Conclusion we assess our work and what remains to be done, as well as presenting some questions for further research.

Chapter 2

Framework

In this section we describe a framework for modeling the activity of an attacker with cryptanalytical capabilities. We introduce a set of expressions which describe how bitstrings may be constructed from randomly generated data using a finite set of deterministic algorithms and present the semantic interpretation for these expressions. We then use these expressions to represent protocols and their executions, as well as the attacker's knowledge about published messages and possibly other bitstrings used to construct those messages.

2.1 Setting

We consider a public network where agents exchange private messages. To prevent an attacker from compromising the security of the network, the agents use security protocols. These protocols specify how the agents construct the messages they publish using cryptographic functions.

In our framework, each message is a finite sequence of bitstrings (which in turn are finite sequences of bits), *i.e.*, an element of $(\{0,1\}^*)^*$. Of course, any finite sequence of bitstrings may be encoded as an element of $\{0,1\}^*$ (the set of bitstrings); for practical reasons which will later become clear, we prefer to use the previous definition. From now on, we will denote the set of sequences of bitstrings by $\mathbf{B} = (\{0,1\}^*)^*$. Again for practical reasons, we will often refer to elements of $\mathbf{B}$ simply as bitstrings.

To represent the functions used by the agents to execute the protocols, we use a finite set $\mathcal{F}$ of deterministic algorithms and a set $\mathcal{R}$ of probabilistic algorithms. Since we want to model an attacker A with additional cryptanalytical capabilities, we consider another set of deterministic algorithms, $\mathcal{F}^A \supseteq \mathcal{F}$, and probabilistic algorithms, $\mathcal{R}^A \supseteq \mathcal{R}$, which A may use to obtain and represent knowledge about the secret data involved in the protocols or to compose messages of his own.

To allow reasoning about complexity issues and ultimately modelling computationally feasible attacks which explore cryptanalysis, we consider that each algorithm depends on a security parameter. Note that it is possible to simulate the execution of a general probabilistic algorithm by means of a deterministic algorithm and a probabilistic algorithm which represents the randomness involved in the calculations and depends only on the security

parameter. For this reason, we will assume (without loss of generality) that all algorithms in the sets $\mathcal{R}, \mathcal{R}^A$ receive only the security parameter as input. We will often refer to such algorithms as random generation algorithms.

Thus, algorithms in $\mathcal{R}$ are random generation algorithms which depend only on the security parameter. Deterministic algorithms receive this parameter and any finite sequence of bitstrings $b \in \mathbf{B}$. The output of these algorithms is a finite sequence of bitstrings $b' \in \mathbf{B}$.

2.2 Expressions

Each message in our framework is constructed inductively by applying deterministic algorithms to randomly generated bitstrings and possibly fixed bitstrings chosen by the agents. Thus, we represent the construction of each message by an expression involving:

- the symbols $F \in \mathcal{F}^A$ to represent the execution of the corresponding (deterministic) algorithms by either the attacker A or honest agents;
- symbols representing an execution of a random generation algorithm;
- bitstrings.

Each random generation algorithm $R \in \mathcal{R}^A$ may be executed several times. Since each of these executions may output a different bitstring, we need to distinguish between them. Thus, for each $j \in \mathbb{N}$, the expression R^j represents a different execution of the random generation algorithm R. These considerations justify the following definition.

Definition 2.2.1. *Let $\mathcal{G}$ be a set of deterministic algorithms and $\mathcal{S}$ a set of random generation algorithms. The set $\mathbf{Exp}(\mathcal{G}, \mathcal{S})$ of expressions generated by the sets $\mathcal{G}, \mathcal{S}$ is defined inductively as follows:*

- *for each $j \in \mathbb{N}$ and $R \in \mathcal{S}$, $R^j \in \mathbf{Exp}(\mathcal{G}, \mathcal{S})$;*
- *$\mathbf{B} \subseteq \mathbf{Exp}(\mathcal{G}, \mathcal{S})$;*
- *if $E_1, \ldots, E_n \in \mathbf{Exp}(\mathcal{G}, \mathcal{S})$ and $F \in \mathcal{G}$, $F(E_1, \ldots, E_n) \in \mathbf{Exp}(\mathcal{G}, \mathcal{S})$.*

When the sets $\mathcal{F}, \mathcal{R}$ of public algorithms and $\mathcal{F}^A, \mathcal{R}^A$ of algorithms available to an attacker A are clear from the context, we abbreviate $\mathbf{Exp} = \mathbf{Exp}(\mathcal{F}, \mathcal{R})$, $\mathbf{Exp}^A = \mathbf{Exp}(\mathcal{F}^A, \mathcal{R}^A)$.

Note that in the previous definition we use the same symbol F in two different contexts: as part of the representation of an expression $E \in \mathbf{Exp}$ and to represent the function calculated by the algorithm F. We use this notation throughout this thesis; the context should avoid ambiguities.

Intuitively, each expression in $\mathbf{Exp}$ (resp. $\mathbf{Exp}^A$) represents the construction of a message using only public algorithms (resp. all the algorithms available to the attacker A).

Suppose that we fix a set of deterministic algorithms $\mathcal{F}$ and a set of probabilistic algorithms $\mathcal{R}$. For each security parameter η, there is a natural way to interpret each expression $E \in \mathbf{Exp}$ as a random variable E_η, whose possible values are bitstrings. Intuitively, for any bitstring $b \in \mathbf{B}$, $P[E = b]$ is the probability that b is the bitstring obtained after performing the computations prescribed by E.

We will write $b \leftarrow R$ to denote that b is sampled from the random variable R. Given a random generation algorithm $R \in \mathcal{R}$ and a security parameter η, the output of $R(\eta)$ may be interpreted as a random variable. Thus, for simplicity, we will also write $b \leftarrow R(\eta)$ to denote that b is the output of an execution of the random generation algorithm R for security parameter η. The random variables corresponding to other expressions (involving bitstrings and deterministic algorithms) are defined as follows:

Definition 2.2.2. *Let $E \in \mathbf{Exp}$ be an expression and fix a security parameter $\eta \in \mathbb{N}$. The random variable E_η is defined inductively as follows:*

- *if $R \in \mathcal{R}, j \in \mathbb{N}$, then $R^j_\eta \leftarrow R(\eta)$;*
- *if $b \in \mathbf{B}$, then $b_\eta = b$;*
- *if $E^1, \ldots, E^n \in \mathbf{Exp}$ and $F \in \mathcal{F}$, then*

$$(F(E^1, \ldots, E^n))_\eta = F(\eta, E^1_\eta, \ldots, E^n_\eta).$$

Note that we assume that each execution of the random generation algorithms is independent. As such, if $R \in \mathcal{R}$, $i \neq j \in \mathbb{N}$, then R^i_η, R^j_η are independent random variables with the same probability distribution. Similarly, if $R \neq R^* \in \mathcal{R}$, $i, j \in \mathbb{N}$, then $R^i_\eta, R^{*,j}_\eta$ are also independent random variables.

In order to obtain the probability distribution of E_η, we first define the sets of subexpressions and random subexpressions of an expression. This is essentially a list of the executions of random generation algorithms used in the construction of a message represented by E.

Definition 2.2.3. *Let $E \in \mathbf{Exp}$ be an expression. The set of subexpressions (resp. random subexpressions) of E, denoted by $sub(E)$ (resp. $random(E)$), is defined inductively as follows:*

- *if $R \in \mathcal{R}, j \in \mathbb{N}$,*

$$sub(R^j) = \{R^j\}$$

$$(resp.\ random(R^j) = \{R^j\});$$

- *if $b \in \mathbf{B}$,*

$$sub(b) = \{b\}$$

$$(resp.\ random(b) = \{\});$$

- *if* $E_1, \ldots, E_n \in \mathbf{Exp}$ *and* $F \in \mathcal{F}$,

$$sub(F(E_1, \ldots, E_n)) = \{F(E_1, \ldots, E_n)\} \cup \bigcup_{i=1}^{n} sub(E_i)$$

$$(resp.\ random(F(E_1, \ldots, E_n)) = \bigcup_{i=1}^{n} random(E_i)).$$

This definition is useful because the bitstring corresponding to an expression depends only on the result of the random generation algorithms involved in the expression.

Given expressions $E, S, S' \in \mathbf{Exp}$, we define $E[S'/S]$ to be the expression obtained from E by substituting all occurrences of S by S'.

Using this definition, we will now introduce the concept of bitstring represented by an expression without random subexpressions for a security parameter η. Intuitively, if an expression has no random subexpressions, it is composed only of bitstrings and computations of deterministic algorithms; thus, for each security parameter η, we may compute its corresponding value. This concept will be useful to define the distribution of probability of a general expression (which of course may include random subexpressions).

Definition 2.2.4. *Fix* $\eta \in \mathbb{N}$ *and let* $E \in \mathbf{Exp}$ *be such that* $random(E) = \{\}$. *We write* b_η *for the bitstring represented by* E *for* η, *which we define inductively as follows:*

- *if* $E = b \in \mathbf{B}$, *then*

$$b_\eta(E) = b;$$

- *if* $E = F(M_1, \ldots, M_n)$ *for some* $F \in \mathcal{F}, M_1, \ldots, M_n \in \mathbf{Exp}$, *then*

$$b_\eta(E) = F(\eta, b_\eta(M_1), \ldots, b_\eta(M_n)).$$

Using our assumption that the output of each execution of a random generation algorithm is independent of everything else, we may obtain the distribution of E_η in terms of the probability distribution of the outputs of each random generation algorithm $R \in \mathcal{R}$ for the security parameter η. Writing $random(E) = \{R_1^{j_1}, \ldots, R_n^{j_n}\}$ and abbreviating $[b_i/R_i^{j_i}]$ by σ_i for each $i = 1, \ldots, n$, the probability distribution of E_η is given by

$$\begin{aligned} P[E_\eta = b] &= P[b_\eta(E\sigma_1 \ldots \sigma_n) = b \mid b_1 \leftarrow R_1(\eta), \ldots, b_n \leftarrow R_n(\eta)] \\ &= \sum_{\substack{b_1, \ldots, b_n : \\ b_\eta(E\sigma_1 \ldots \sigma_n) = b}} P[R_1(\eta) = b_1, \ldots, R_n(\eta) = b_n] \\ &= \sum_{\substack{b_1, \ldots, b_n : \\ b_\eta(E\sigma_1 \ldots \sigma_n) = b}} \prod_{k=1}^{n} P[R_k(\eta) = b_k]. \end{aligned} \tag{2.1}$$

Thus, for each security parameter η and each $R \in \mathcal{R}$, each expression $R^j, j \in \mathbb{N}$ represents an independent random variable R^j_η whose probability distribution is equal to the probability distribution of the outputs of $R(\eta)$. Observe that in the last equation the function b_η is used properly, since $random(E) = \{R^{j_1}_1, \ldots, R^{j_n}_n\}$ and thus $random(E\sigma_1 \ldots \sigma_n) = \{\}$.

2.3 Representing Protocols

Now we describe how security protocols may be represented in our framework. We assume that the attacker is a legitimate user of the network, and thus may participate in the execution of the protocols, but unlike other users he does not need to respect the specifications of the protocols and may use algorithms other than the public algorithms of the environment.

Several protocols may be used in the environment. At any given time, several sessions of each protocol may (in principle) be in execution, each of which may be in any point of its execution. We assume that the attacker completely knows the structure of each message published, *i.e.*, which expression $E \in \mathbf{Exp}$ represents the construction of the message. This is reasonable if we assume that all other agents are honest and follow the protocols: in this case, the attacker knows how each agent constructs each message (using public deterministic and random generation algorithms) from the messages published in the network. We also assume that all agents know which agent published each message, but, as in the standard Dolev-Yao model, we allow our attacker to impersonate other agents.

We represent the publication of a message as part of the execution of a protocol by a tuple (P, i, s, A, E), which means that the s-th step of the execution of the session i of the protocol P, agent A published a message whose construction is represented by an expression E. By "step" of an execution of a protocol we mean the publication of a message; as such, the s-th step of the execution of a protocol is the publication of the s-th message. Thus,

- P is the protocol;
- $i \in \mathbb{N}$ is an identifier for the session of the protocol P;
- $s \in \mathbb{N}$ is the step of the execution of the session for which the message was sent;
- $A \in \{0,1\}^*$ is the name of the agent which published the message;
- $E \in \mathbf{Exp}$ is the expression which represents the construction of the message from previously exchanged messages and random data generated by the agents.

We use a set $\mathcal{H}$ of tuples of this form to record the publication of each mesage in the execution of protocols. We use this set to represent the specification of protocols in our framework, by means of a set of rules which specify how new tuples may be added to the set $\mathcal{H}$.

We first present a simple example to illustrate this principle. The following is a simplified representation of the Needham-Schroeder public key protocol in our framework, which in particular does not include the decryption function. In the next sections we present a more complete description, including the representation of the encryption scheme.

Example 2.3.1 (The Needham-Schroeder Public Key Protocol)**.** *We consider the sets of public algorithms* $\mathcal{F} = \{K, P, enc\}$ *and* $\mathcal{R} = \{R\}$*, where:*

- K *is such that, for each agent name* a [1]*,* $K(a)$ *returns* a*'s public key;*
- P *is a pairing function; for simplicity, we will write* $P(E_1, \ldots, E_n) \equiv (E_1, \ldots, E_n)$*;*
- enc *is the encryption function,* i.e.*,* $enc(k, m)$ *is the encryption of message* m *with public key* k*; again for simplicity, we write* $enc(k, m) \equiv \{m\}_k$*;*
- R *is a public random generation algorithm.*

In the standard Alice&Bob-notation, the protocol is represented as follows:

$$\begin{aligned} A &\to B : \{A, R^1\}_{K(B)} \\ B &\to A : \{R^1, R^2\}_{K(A)} \\ A &\to B : \{R^2\}_{K(B)} \end{aligned} \tag{2.2}$$

The following three rules for adding elements to the set $\mathcal{H}$ *compose a (simplified) representation of the* $NSPK$ *protocol in our framework:*

- $$\frac{}{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 1, A, \{(A, R^j)\}_{K(B)})\}}, \quad i, j \text{ "fresh"}$$
- $$\frac{(NSPK, i, 1, A, \{(A, E)\}_{K(B)}) \in \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 2, B, \{(E, R^i)\}_{K(A)})\}}, \quad i \text{ "fresh"}$$
- $$\frac{(NSPK, i, 1, A, \{(A, E)\}_{K(B)}), (NSPK, i, 2, B, \{(E, E')\}_{K(A)}) \in \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 3, A, \{E'\}_{K(B)})\}}$$

In the second step, the user B *may be responding to any pair* $(A, E)_{K(B)}$ *instead of only to pairs* $(A, R_i)_{K(B)}$*. This is because he may not know how* E *was generated, and even if the message was manipulated he will still assume that it was honestly generated using algorithm* R*. Similar considerations justify the third rule. Note that using a more complex representation we could model a situation in which agent* B *may even respond to messages which have not been encrypted with his private key or whose underlying plaintext is not a pairing of messages.*

[1] We tacitly assume that agent names are bitstrings: *i.e.*, $a \in \mathbf{B}$ for each agent name a.

In general, to describe a protocol P, we use the rules which determine how to construct the message corresponding to each step s of the execution of the protocol and which agent should publish it. Both the message and the agent may depend on the previous $s-1$ steps.

Let E_i be the expression representing the message published in the i-th step of the execution of the protocol and A_i be the agent who published it. We assume that there is a computable function S which, given the history of the execution of protocols, tells the agents what they can do next. That is, for any s, $S((A_1, E_1), \ldots, (A_{s-1}, E_{s-1}))$ is the set of pairs (A, E) such that agent A sending a message represented by E is a valid s-th step for the protocol. Of course, S need not be total; the execution of the protocol may terminate, or the agents may not respond if the form of the messages is not correct. Also note that S may depend on other inputs, such as the security parameter.

Thus, the valid rules for the protocol P are the rules of the form

$$\frac{(P, i, 1, A_1, E_1), \ldots, (P, i, s-1, A_{s-1}, E_{s-1}) \in \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(P, i, s, A, E)\}}, \tag{2.3}$$

where:

- s is the number of the step
- i is any natural number representing a session of the protocol
- $A, A_1, \ldots, A_{s-1}$ are agent names
- $E, E_1, \ldots, E_{s-1} \in \mathbf{Exp}$
- $(A, E) \in S((A_1, E_1), \ldots, (A_{s-1}, E_{s-1}))$.

There are a number of situations in which the rules above may not be the best way to represent a particular setting. For example, consider an environment in which, if the messages corresponding to the first s steps of some execution of a protocol have already been published, then no honest agent may publish a message corresponding to the previous steps $1, \ldots, s-1$.

The rules described above do not account for this possibility: as long as the first $t-1$ messages have been published, an agent may always publish a valid message for the t step of the protocol, even if messages $t+1, t+2, \ldots$ have already been published as well. This may allow the attacker to read several different (and valid) messages for the t-th step of the protocol. He would then be obtaining more information than he would have obtained in a real situation, in which the agents would not have repeatedly published messages corresponding to the t-th step of the protocol.

To model this situation we need to add another premise to the rules presented above, thus obtaining:

$$\frac{(P,i,1,A_1,E_1),\ldots,(P,i,s-1,A_{s-1},E_{s-1}) \in \mathcal{H} \quad j > s \Rightarrow (P,i,j,A_j,E_j) \notin \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(P,i,s,A,E)\}}, \tag{2.4}$$

where A_j is any agent and E_j is any expression. Observe that we allow multiple messages corresponding to the s-th step of the protocol to be "published", as long as no message corresponding to a step $t > s$ is added to the set $\mathcal{H}$. This is because the attacker may intercept messages and prevent them to be delivered. Intercepting messages corresponding to a given step is a valid action; however, once a message corresponding to a step $t > s$ appears in $\mathcal{H}$, then the s-th message must have been read and responded to by the agents. Thus, he may not expect any agent to execute the s-th step of that session of the protocol again.

2.4 The Attacker

2.4.1 Knowledge of the Attacker

We represent the attacker's knowledge by a set $\mathcal{K}$ of statements of the form

$$E = b, \tag{2.5}$$

where $E \in \mathbf{Exp}, b \in \mathbf{B}$. Such a statement implies that all random subexpressions of E have been sampled from the corresponding random generation algorithm during the execution of the protocol. In this case, $E_\eta = b$ is effectively the bitstring obtained by substituting in E each random subexpression by the corresponding randomly generated bitstring and performing the remaining deterministic computations prescribed by E using some security parameter η fixed for the environment (cf. Definition 2.2.2).

We now present two standard ways in which the knowledge of the attacker may increase. As usual, we denote the security parameter in use in the environment by η.

- Reading a published message. When a message whose construction is described by $E \in \mathbf{Exp}^A$ is published, all random subexpressions of E must have been sampled from the corresponding random generation algorithms during the execution of the protocol, thus determining E_η. Following this remark, reading a message is a method of increasing the attacker's knowledge described by the following rule:

$$\frac{(P,i,s,A,E) \in \mathcal{H}}{\mathcal{K} := \mathcal{K} \cup \{E = E_\eta\}}. \tag{2.6}$$

- Running an algorithm. This models the computation of the value of some random generation algorithm $R \in \mathcal{R}^A$ or some deterministic algorithm $F \in \mathcal{F}^A$ on expressions

$E_1, \ldots, E_n$ whose corresponding bitstrings he knows. Formally,

$$\frac{(E_i = b_i) \in \mathcal{K} \text{ or } E_i \in \mathbf{B}, E_i = b_i \text{ for each } i = 1, \ldots, n}{\mathcal{K} := \mathcal{K} \cup \{F(E_1, \ldots, E_n) = F(\eta, b_1, \ldots, b_n)\}}, F \in \mathcal{F}^A \tag{2.7}$$

$$\frac{}{\mathcal{K} := \mathcal{K} \cup \{R^i = r\}}, R \in \mathcal{R}^A, r \leftarrow R(\eta), i \text{ fresh.} \tag{2.8}$$

In the next sections we discuss other ways for the attacker's knowledge to increase.

2.4.2 Participation of the Attacker in Protocols

The attacker may participate in the protocols as an honest agent, but may also impersonate other agents or publish messages constructed by ways other than that specified by the protocol. Since it may be useful to distinguish between messages sent by the attacker from messages sent by other agents, we will represent a message sent by the attacker A impersonating agent B by adding an exponent A to the tuple used to represent the publication of messages, thus obtaining $(P, i, s, B, E)^A$. Unless stated otherwise, such messages are interpreted just as other messages for the purpose of interpreting the rules which describe how the sets $\mathcal{H}$ and $\mathcal{K}$ may grow. In particular, the attacker may always participate honestly in a protocol. The messages honestly published by an attacker A are followed by an $\_^A$; the agent name associated with those messages is his own.

The general rule to represent the participation of the attacker in a protocol is

$$\frac{\begin{array}{c}(P, i, 1, A_1, E_1), \ldots, (P, i, s-1, A_{s-1}, E_{s-1}) \in \mathcal{H} \\ (E = b) \in \mathcal{K}\end{array}}{\mathcal{H} := \mathcal{H} \cup \{(P, i, s, B, E)^A\}}. \tag{2.9}$$

The second condition $(E = b) \in \mathcal{K}$ means that he may only publish the message corresponding to an expression E if he knows the bitstring represented by that message. Note that he may use any expression E he knows and impersonate any agent B. Of course, if this participation is not valid, the protocol execution may stop, and the attacker may not profit from his intrusion.

2.4.3 Actions, Attacks, Strategies and Complexity

We will say that an "action" of the attacker is the application of any rule which adds one element to either the set $\mathcal{K}$ or the set $\mathcal{H}$. We have already described above some of these rules; we will present more in the following sections.

Observe that, according to this definition, the publication of a message by any honest agent is an action of the attacker, since the set $\mathcal{H}$ increases. The reason for this is that we want the protocols to be safe for any possible sequence of its executions. Thus, it is a valid action of the attacker to wait for some protocol step to be executed. An attack

which depends on the execution of some particular step of a protocol is still a successful and potentially dangerous attack.

An "attack" is a sequence of valid actions at the end of which the sets $\mathcal{H}, \mathcal{K}$ have some predetermined "bad" property. Exactly what these properties are may vary greatly depending on the protocol and its goals. For example, a successful attack may be finding the bitstring corresponding to some expression. For another example, an attack to the Needham-Schroeder public key protocol (in the simplified version presented in 2.3.1) is completing an execution of the protocol impersonating some other agent. This is represented by one of two conditions:

$$\begin{array}{c} (NSPK, i, 1, B, \{(B, E)\}_{K(C)})^A \\ (NSPK, i, 2, C, \{(E, R^i)\}_{K(B)}) \quad \in \mathcal{H}, \text{ or} \\ (NSPK, i, 3, B, \{R^i\}_{K(C)})^A \end{array} \tag{2.10}$$

$$\begin{array}{c} (NSPK, i, 1, C, \{(B, R^i)\}_{K(B)}) \\ (NSPK, i, 2, B, \{(R^i, E)\}_{K(C)})^A \quad \in \mathcal{H}. \\ (NSPK, i, 3, C, \{E\}_{K(B)}) \end{array} \tag{2.11}$$

Here, B, C are the names of some agents other than the attacker and $E \in \mathbf{Exp}$ is any expression. Both of the conditions presented represent executions in which the attacker impersonates an agent B in protocol executions with agent C. In the first case the attacker is the initiator, while in the second case is the responder. Note that B, C may be any agents.

A "strategy of attack" is an algorithm, possibly probabilistic, which, given the sets $\mathcal{H}$, $\mathcal{K}$ and the security parameter η, returns a valid action for the attacker. Intuitively, given the history of the execution of protocols in the environment and the current knowledge of the attacker, a strategy will determine the next step of the attacker, which may be any of the valid actions which we have already discussed (waiting for some step of a protocol to be executed, publishing or reading a message and computing the value of some function he knows on expressions whose corresponding bitstrings he knows) or others which we will describe in the next sections. This function need not be defined for all possible sets $\mathcal{H}$ and $\mathcal{K}$: for some states of execution, the attacker may have finished his attack or simply given up, and thus no next action is defined. It is worth noting that such a definition makes our model of strategy an operational one, as opposed to a denotational one.

In the following, we will frequently describe strategies of attack as algorithms for the attacker. Such algorithms describe a sequence of steps which the attacker may perform in sequence, instead of an attack strategy as we have defined it above. However, given one such algorithm it is simple to define an attack strategy (*i.e.*, a function which given η, $\mathcal{H}$ and $\mathcal{K}$ chooses the attacker's next action) which represents the same attack, and it is often more convenient to describe a strategy in this fashion.

A final remark on complexity: if we want to model an attacker with limited (polynomial) computational power, we must consider only strategies which involve a number of actions bounded by a polynomial function of the security parameter. Furthermore, all the algorithms

with which we equip the attacker must also be computable in polynomial time, as does the function representing the strategy of attack (the attacker must be able to decide in polynomial time which will be his next action). If all these conditions are true, then the number of operations of the attacker is also polynomial on the security parameter.

Chapter 3

Cryptographic Properties

In this section we describe what cryptographic properties we consider in our framework and present some examples which illustrate how this concept relates to well-known cryptographic weaknesses such as malleability. We describe how these properties may be used to find attacks using our framework. Finally, we show how indistinguishability under chosen plaintext attacks (a standard security property of encryption schemes) may be expressed in terms of these properties.

3.1 Definition and Basic Examples

In all generality, a cryptographic property is simply a probabilistic relation between expressions. This means that, knowing the bitstrings corresponding to certain expressions, the attacker is able to learn "something" about the bitstrings corresponding to other expressions. More precisely:

Definition 3.1.1. *A cryptographic property is a statement of the form*

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*] = p, \tag{3.1}$$

where $E_1, \ldots, E_n, E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{Exp}, b_1, \ldots, b_n, b_1^*, \ldots, b_{n^*}^* \in \mathbf{B}$ *and* $p \in [0, 1]$.

Recall that, if E is an expression, E_η is the random variable whose value is the bitstring obtained by constructing a message in the way described by E and using security parameter η in the computations.

In the following we will frequently discuss asymmetric encryption schemes as important examples. Let us then define this concept in our framework.

Definition 3.1.2. *An asymmetric encryption scheme is a tuple of algorithms*

$$(R_k, pub, priv, R_e, Enc, Dec), \tag{3.2}$$

where:

- $R_k \colon \mathbb{N} \to \mathbf{B}$ *is probabilistic: it receives a security parameter* $\eta \in \mathbb{N}$ *and returns a key*

valid for that security parameter. We write

$$\mathbf{Keys}(\eta) = \{b \in \mathbf{B} \colon P[R_k(\eta) = b] > 0\}$$

to denote the set of valid keys for security parameter η.

- *$pub, priv \colon \mathbb{N} \times \mathbf{B} \to \mathbf{B}$ are deterministic: they receive a security parameter $\eta \in \mathbb{N}$ and a valid key for that security parameter, and return the corresponding public (resp. private) key. We write*

$$\mathbf{PubKeys}(\eta) = pub(\mathbf{Keys}(\eta)),$$
$$\mathbf{PrivKeys}(\eta) = priv(\mathbf{Keys}(\eta))$$

to denote the set of valid public (resp. private) keys for security parameter η.

- *$R_e \colon \mathbb{N} \to \mathbf{B}$ is probabilistic: it receives a security parameter $\eta \in \mathbb{N}$ and returns a sequence of bitstrings which may be used to model the randomness involved in the encryption algorithm Enc for η. Let*

$$\mathbf{Random}(\eta) = \{b \in \mathbf{B} \colon P[R_e(\eta) = b] > 0\}.$$

- *$Enc \colon \bigcup_{\eta \in \mathbb{N}}(\{\eta\} \times \mathbf{Plain}(\eta) \times \mathbf{PubKeys}(\eta) \times \mathbf{Random}(\eta)) \to \mathbf{B}$, where $\mathbf{Plain}(\eta)$ is the set of valid plaintexts for security parameter η, is the encryption algorithm. Note that even if the encryption scheme is probabilistic, the inclusion of a randomly generated argument allows us to describe it using the deterministic algorithm Enc. We write*

$$\mathbf{Cipher}(\eta) = Enc(\eta, \mathbf{Plain}(\eta), \mathbf{PubKeys}(\eta), \mathbf{Random}(\eta))$$

to denote the set of valid ciphertexts for security parameter η.

- *$Dec \colon \bigcup_{\eta \in \mathbb{N}}(\{\eta\} \times \mathbf{Cipher}(\eta) \times \mathbf{PrivKeys}(\eta)) \to \mathbf{B}$ is the deterministic decryption algorithm.*

Furthermore, the decryption function is the "inverse" of the encryption: i.e., *for each security parameter $\eta \in \mathbb{N}$, key $k \in \mathbf{Keys}(\eta)$, plaintext $p \in \mathbf{Plain}(\eta)$ and random bitstring $r \in \mathbf{Random}(\eta)$,*

$$Dec(\eta, Enc(\eta, p, pub(k), r), priv(k)) = p. \tag{3.3}$$

We now present some remarks about our definition of cryptographic property.

Remark 3.1.3. *(i) Not all cryptographic properties are weaknesses: indeed, certain cryptographic properties are necessary for the cryptographic primitives to work properly. For example, if $(R_k, pub, priv, R_e, Enc, Dec)$ is an asymmetric encryption scheme, we always have*

$$P[(Dec(Enc(E, pub(R_k), R_e), priv(R_k)))_\eta = b \mid E_\eta = b] = 1, \tag{3.4}$$

$$P[E_\eta = b \mid (Dec(Enc(E, pub(R_k), R_e), priv(R_k)))_\eta = b] = 1. \tag{3.5}$$

(ii) Any deterministic or random generation algorithm may be completely described in terms of these properties.

If F is a deterministic algorithm it is determined by considering a set of properties of the form

$$P[(F(E^1, \ldots, E^n))_\eta = F(\eta, b_1, \ldots, b_n) \mid E^1_\eta = b_1, \ldots, E^n_\eta = b_n] = 1 \tag{3.6}$$

for all security parameters η and all bitstrings $b_1, \ldots, b_n$ such that, for each i, $P[E^i_\eta = b_i] > 0$.

If R is a random generation algorithm, it is described by considering the set of properties of the form

$$P[R_\eta = b] = p \tag{3.7}$$

for all security parameters η and all bitstrings b.

(iii) Similarly, suppose that for all bitstrings $b \in \mathbf{B}$, expressions $X, Y \in \mathbf{Exp}^A$ and security parameters $\eta \in \mathbb{N}$ we have

$$P[E_\eta = b' \mid (Enc(E, K, Y))_\eta = b] = 1. \tag{3.8}$$

Of course, it is possible that the attacker knows some properties of this form (for example by encrypting b' with a public key represented by K and random data represented by Y. However, if we equip the attacker with similar properties for "too many" bitstrings $b \in \mathbf{B}$ and expressions $X, Y \in \mathbf{Exp}^A$, we may be modelling an attacker irrealistically powerful: in particular, if he knows these properties for all b, X, Y, he may completely break the encryption scheme. This example shows that if we want to model an attacker with limited computational power we need to be careful regarding the limitations we impose to his knowledge.

(iv) We may use these properties to represent the leakage of partial information. For example, we may represent properties like

> *"If the last bit of the encryption of a message is 0, then the last bit of the underlying plaintext is 0 with probability 0.6."*

writing

$$P[(l(E))_\eta = 0 \mid (l(Enc(E, K)))_\eta = 0] = 0.6,$$

where l is a polynomial-time algorithm which, given a bitstring, returns it's last bit.

Such properties may be useful, for example, when describing attacks which use linear and differential cryptanalysis.

In order to model the attacker's knowledge about the cryptography, we assume that he knows some cryptographic properties. More precisely, for some expressions $E^1, \ldots, E^n$,

$E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{Exp}$ and some bitstrings $b_1, \ldots, b_n,\ b_1^*, \ldots, b_{n^*}^* \in \mathbf{B}$ the attacker is able to compute

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*];$$

for other expressions and other bitstrings he may not be able to compute this probability.

For this we consider that he has available a function

$$p\colon \mathbb{N} \times ((\mathbf{Exp} \times \mathbf{B})^*)^2 \to [0,1] \cup \{\bot\} \tag{3.9}$$

such that

$$\begin{array}{c} p[\eta, (((E^1, b_1), \ldots, (E^n, b_n)), ((E^{*,1}, b_1^*), \ldots, (E^{*,n^*}, b_{n^*}^*)))] = \rho \in [0,1] \\ \Rightarrow P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*] = \rho. \end{array} \tag{3.10}$$

Intuitively, the attacker "asks" p for the probability

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*];$$

p either returns the correct probability or $\bot$ otherwise.

For simplicity, we will write

$$p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] \tag{3.11}$$

instead of

$$p[\eta, (((E^1, b_1), \ldots, (E^n, b_n)), ((E^{*,1}, b_1^*), \ldots, (E^{*,n^*}, b_{n^*}^*)))]. \tag{3.12}$$

The problem in remark 3.1.3.(iii) may be solved, for example, if we require that p is computable in polynomial time (as a function of η). However, it may also be interesting to consider other cases. For example, we may want to study the impact of some cryptographic properties on the security of a given protocol, even if it is not known whether the cryptographic primitives verify those properties or whether there is an algorithm for calculating the resulting oracle p in polynomial time.

Realistically, the attacker may not be able to obtain the exact value of these probabilites and still find a reasonable estimation. Since our goal will be to obtain an estimation of the probability distribution of random variables represented by expressions given the knowledge of the attacker, we only need to perform our calculations using approximate values instead of exact ones. Studying the error of our estimations in terms of the precision of the attacker's knowledge is an interesting problem (even if the attacker knows exact values) but is beyond the scope of this work.

Remark 3.1.3.(i) justifies allowing the attacker to use another type of action, which may depend on the particular security parameter η being used:

$$\frac{\begin{array}{c}(E^{*,1} = b'_1), \ldots, (E^{*,n^*} = b^*_{n^*}) \in \mathcal{K} \\ p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b^*_1, \ldots, E^{*,n^*} = b^*_{n^*}] = 1\end{array}}{\mathcal{K} := \mathcal{K} \cup \{E^i = b_i\}}, \quad i = 1, \ldots, n. \tag{3.13}$$

Intuitively, this rule allows the attacker to combine his knowledge of the algorithms being used (described by the function p) and the messages being exchanged in the environment (described by the set $\mathcal{K}$) to learn something new about the published messages. A simple and meaningful example of the use of this rule is the asymmetric encryption scheme. In this case, the properties (3.4), (3.5) allows the attacker to use the rules

$$\frac{(Dec(Enc(E, pub(R_k), R_e), priv(R_k))) = b \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{E = b\}}, \tag{3.14}$$

$$\frac{E = b \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{(Dec(Enc(E, pub(R_k), R_e), priv(R_k))) = b\}}. \tag{3.15}$$

We now present two examples which further illustrate how these properties may be used to model the knowledge and activity of the attacker. We will use these examples to represent an attack discovered by Warinschi to the Needham-Schroeder-Lowe protocol.

In the first example we describe two natural "congruence" rules and show how to represent them in our framework.

Example 3.1.4. *Let $E, S, S' \in$ **Exp**. Recall that we denote by $E[S'/S]$ the expression obtained from E by substituting all occurrences of the subexpression S by S'.*

It is easy to see that

$$P[(E[S'/S])_\eta = b \mid S_\eta = b', S'_\eta = b', E_\eta = b] = 1 \tag{3.16}$$

*always holds. This is a congruence rule: if two sampled expressions S, S' correspond to the same bitstring (*i.e.*, $S_\eta = S'_\eta = b'$), we may substitute all occurrences of S by S' in any expression E without changing the bitstring represented by E. Thus, $E_\eta = (E[S/S'])_\eta$, and we obtain*

$$p_\eta[E[S'/S] = b \mid S = b', S' = b', E = b] = 1, \tag{3.17}$$

which may be interpreted as a substitution rule and is always true. Hence, by 3.13, we may allow the attacker to increase his knowledge by using the rule

$$\frac{(S = b'), (S' = b'), (E = b) \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{(E[S'/S] = b)\}}. \tag{3.18}$$

Now suppose that for some $S, S' \in \mathbf{Exp}$ *and all* $\eta \in \mathbb{N}, b \in \mathbf{S}$ *we have*

$$\begin{array}{l} P[S'_\eta = b \mid S_\eta = b] = 1 \\ P[S_\eta = b \mid S'_\eta = b] = 1 \end{array} \tag{3.19}$$

*whenever the expressions make sense (*i.e.*, when* $P[S = b] > 0, P[S' = b] > 0$*).*

It is easy to see that these properties are equivalent to

$$P[S_\eta = S'_\eta] = 1 \text{ for all } \eta \in \mathbb{N}. \tag{3.20}$$

This holds, for example, for $S' = Dec(Enc(E, pub(R_k), R_e), priv(R_k)), S = E$ *(cf.* (3.4), (3.5)*).*

If such a property holds, then the function p which represents the cryptanalytical knowledge of the attacker should be such that

$$\begin{array}{l} p_\eta[E = b \mid E[S'/S] = b] = 1, \\ p_\eta[E[S'/S] = b \mid E = b] = 1 \end{array} \tag{3.21}$$

for any E, η*. Thus, by* (3.13)*, we obtain the rules:*

$$\frac{((E[S'/S])_\eta = b) \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{E = b\}}, \tag{3.22}$$

$$\frac{(E_\eta = b) \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{E[S'/S] = b\}}. \tag{3.23}$$

In both cases presented here the reasoning would still be valid if we replaced only some (instead of all) occurrences of a given expressions by another expression representing the same bitstring. To simplify the exposition we do not consider such cases here.

In the next example we describe the Elgamal encryption scheme. We also present its malleability property and show how to represent it in our framework. In the following we will assume that there is an efficient algorithm for converting natural numbers into bitstrings (and vice-versa). As such, we will use natural numbers as arguments of algorithms just as if they were bitstrings.

Example 3.1.5. *Let* $(G_\eta)_{\eta \in \mathbb{N}}$ *be a sequence of "suitable" cyclic groups of bitstrings, and denote the group operation of* G_η *applied to* a *and* b *by* $(a \cdot_\eta b)$*. The Elgamal encryption scheme for these groups is described in our framework by the tuple* $(keygen, pub, priv, encgen, enc, dec)$ *such that, for each* η*:*

- *for each possible value* (g, x) *of* $keygen(\eta)$*,* g *is a generator of the cyclic group* G_η *and* $x \in \{0, \ldots, |G_\eta|\}$*;*
- $pub(\eta, (g, x)) = (g, g^x)$ *generates a public key using random data* (g, x)*;*

- *$priv(\eta,(g,x))=x$ generates a private key using random data (g,x);*
- *for each possible value x of $encgen(\eta)$, $x\in\{0,\ldots,|G_\eta|\}$;*
- *$enc(\eta,p,k,y)=(k_1^y, p\cdot_\eta k_2^y)$ encrypts a plaintext $p\in G_\eta$ using the public key $k=(k_1,k_2)$ and data y generated by encgen;*
- *$dec(\eta,c,k)=c_2\cdot_\eta c_1^{-k}$ decrypts a ciphertext $c\in G_\eta^2$.*

Considering an additional algorithm mult (available only to the attacker) such that, for each η, $mult(\eta,a,b)\equiv a\cdot_\eta b$, we see the well-known malleability property of Elgamal:

$$Enc(mult(E,E'),k,y)=mult(E,Enc(E',k,y)). \tag{3.24}$$

Following the reasoning in 3.1.4, we obtain the following rules (a particular case of (3.22), (3.23)*):*

$$\frac{(E[S'/S]=b)\in\mathcal{K}}{\mathcal{K}:=\mathcal{K}\cup\{E=b\}} \tag{3.25}$$

$$\frac{(E=b)\in\mathcal{K}}{\mathcal{K}:=\mathcal{K}\cup\{E[S'/S]=b\}}, \tag{3.26}$$

where E is any expression and $S=Enc(e\cdot e',k,y), S'=e\cdot Enc(e',k,y)$ (or vice-versa) for some expressions e,e'.

In 3.2 we use these rules to illustrate how attacks may be represented in our framework by presenting the famous man-in-the-middle attack to the Needham-Schroeder public key protocol [20] and a less famous attack to the Needham-Schroeder-Lowe protocol discovered by Warinschi [25], which uses a simple cryptographic property.

3.2 Attacks to the NSPK and NSL protocols

First we describe the sets $\mathcal{F},\mathcal{R}$ which we will consider. Since these protocols involve an asymmetric encryption scheme, we need deterministic algorithms $pub, priv, Enc, Dec\in\mathcal{F}$, and random generation algorithms $R_k, R_e\in\mathcal{R}$. We will also use a "pairing" algorithm[1] $pair\in\mathcal{F}$ and its "inverse" $pinv\in\mathcal{F}$. Finally, we need a random generation algorithm $R\in\mathcal{R}$. Recall that agent names are represented by bitstrings.

For every $\eta\in\mathbb{N}, b, b_1,\ldots,b_n\in\mathbf{B}$, these algorithms verify

$$pinv(\eta,pair(b_1,\ldots,b_n),j)=b_j, \tag{3.27}$$

$$Dec(\eta,Enc(\eta,m,pub(k),r),priv(k))=m. \tag{3.28}$$

[1] "Pairing" is a loose term. For simplicity, we will use the same algorithm for building a message from any number j of bitstrings.

Thus, the function p with which we equip the attacker should contain the information in equations (3.21), where either

$$\begin{aligned} S &= pinv(pair(E^1, \ldots, E^n), j) \\ S' &= E^j \end{aligned} \tag{3.29}$$

or

$$\begin{aligned} S &= Dec(Enc(E, Pub(K), R), Priv(K)) \\ S' &= E. \end{aligned} \tag{3.30}$$

In these equations, $E, E^1, \ldots, E^n$, $K, R \in \mathbf{Exp}$, and $j \in \mathbb{N}$.

From this we obtain four new valid rules which the attacker may use to increase his knowledge, as in (3.22), (3.23). The attacker may also obtain his randomly generated key and every other user's public key. Denoting by I (from intruder) the bitstring representing the attacker's name, we thus obtain:

$$\overline{\mathcal{K} := \mathcal{K} \cup \{R_k^I = (R_k^I)_\eta\}}, \tag{3.31}$$

and, for each agent name A,

$$\overline{\mathcal{K} := \mathcal{K} \cup \{pub(R_k^A) = (pub(R_k^A))_\eta\}}. \tag{3.32}$$

We abbreviate the pairing function as we have done in Example 2.3.1. Since now we consider that the encryption algorithm requires an argument r randomly generated by R_e, we abbreviate $Enc(p, k, r)$ by $\{p\}_{k,r}$.

With these sets $\mathcal{F}, \mathcal{R}$ we may describe the Needham-Schroeder protocol with greater detail than in Example 2.3.1. The protocol may now be described by the following rules:

$$\overline{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 1, A, \{(A, R^j)\}_{pub(R_k^B), R_e^m)}\}}, \quad i, j, m \text{ "fresh"} \tag{3.33}$$

$$\frac{(NSPK, i, 1, A, \{(A, E)\}_{pub(R_k^B), E'}) \in \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 2, B, \{(E, R^j)\}_{pub(R_k^A), R_e^m)}\}}, \quad j, m \text{ fresh} \tag{3.34}$$

$$\frac{\begin{array}{c}(NSPK, i, 1, A, \{(A, E)\}_{pub(R_k^B), E_r}), \\ (NSPK, i, 2, B, \{(E, E')\}_{pub(R_k^A), E'_r}) \in \mathcal{H}\end{array}}{\mathcal{H} := \mathcal{H} \cup \{(NSPK, i, 3, A, \{E'\}_{pub(R_k^B), R_e^m)}\}}, \quad m \text{ fresh} \tag{3.35}$$

We may now describe the famous man-in-the-middle attack to NSPK. This attack does not involve any specific cryptographic property (other than the desired properties of the encryption scheme and the pairing algorithm), and may be described formally in much

simpler frameworks. However, it is still interesting to illustrate the use of our framework.

3.2.1 Löwe's Famous Man-in-the-Middle Attack to the NSPK protocol

As we have noted in 2.4.3, since this is a mutual authentication protocol whose purpose is to assure two agents B and C that they are talking to each other, an attack to the protocol is a complete execution of the protocol in which the attacker impersonates some other agent. Adapting the conditions (2.10), (2.11) to the new sets $\mathcal{F}, \mathcal{R}$, we obtain the following two attack conditions:

$$\begin{array}{l} (NSPK, i, 1, B, \{B, E\}_{pub(R_k^C), E_r})^I \\ (NSPK, i, 2, C, \{E, R^1\}_{pub(R_k^B), R_e^m}) \quad \in \mathcal{H}, \text{ or} \\ (NSPK, i, 3, B, \{R^1\}_{pub(R_k^C), E_r'})^I \end{array} \tag{3.36}$$

$$\begin{array}{l} (NSPK, i, 1, C, \{C, R^1\}_{pub(R_k^B), R_e^m}) \\ (NSPK, i, 2, B, \{R^1, E\}_{pub(R_k^C), E_r})^I \quad \in \mathcal{H}. \\ (NSPK, i, 3, C, \{E\}_{pub(R_k^C), R_e^{m'}}) \end{array} \tag{3.37}$$

As before, B, C are names of some agents other than the attacker and $E, E_r, E_r' \in \mathbf{Exp}^I$ are any expressions. The attacker convinces agent C that he is talking to agent B, playing the role of the initiator in the first case and the responder in the second.

The standard man-in-the-middle attack to the Needham-Schroeder protocol may be decribed in standard notation as follows:

$$\begin{array}{ll} A \to I : \{(A, R^1)\}_{pub(R_k^I)} & \\ & I \to B : \{(A, R^1)\}_{pub(R_k^B)} \\ & B \to I : \{(R^1, R^2)\}_{pub(R_k^A)} \\ I \to A : \{(R^1, R^2)\}_{pub(R_k^A)} & \\ A \to I : \{R^2\}_{pub(R_k^I)} & \\ & I \to B : \{R^2\}_{pub(R_k^I)} \end{array} \tag{3.38}$$

We now show how this attack may be represented in our framework. To do so we describe the sequence of actions of the attacker (recall that in each action the attacker adds an element to either the set $\mathcal{H}$ or the set $\mathcal{K}$). For simplicity, we will write (P, i, S, A, E) instead of $\mathcal{H} := \mathcal{H} \cup \{(P, i, S, A, E)\}$ and $E = b$ instead of $\mathcal{K} := \mathcal{K} \cup \{E = b\}$.

1. $(NSPK, 1, 1, A, \{(A, R^1)\}_{pub(R_k^I), R_e^1})$
2. $\{(A, R^1)\}_{pub(R_k^I), R_e^1} = m_{1,1}$
3. $R_k^I = K_I$

4. $priv(R_k^I) = priv(\eta, K_I) = PK$
5. $Dec(\{(A, R^1)\}_{pub(R_k^I), R_e^1}, priv(R_k^I)) = Dec(\eta, m_{1,1}, PK) = p_1$
6. $(A, R^1) = p_1$
7. $pub(R_k^B) = K_B$
8. $R_e^2 = r_e^2$
9. $\{(A, R^1)\}_{pub(R_k^B), R_e^2} = Enc(\eta, p_1, K_B, r_e^2) = m_{2,1}$
10. $(NSPK, 2, 1, A, \{(A, R^1)\}_{pub(R_k^B), R_e^2})^I$
11. $(NSPK, 2, 2, B, \{(R^1, R^2)\}_{pub(R_k^A), R_e^3})$
12. $\{(R^1, R^2)\}_{pub(R_k^A), R_e^3} = m_{2,2}$
13. $(NSPK, 1, 2, I, \{(R^1, R^2)\}_{pub(R_k^A), R_e^3})^I$
14. $(NSPK, 1, 3, A, R^2_{pub(R_k^I), R_e^4})$
15. $R^2_{pub(R_k^I), R_e^4} = m_{1,3}$
16. $Dec(R^2_{pub(R_k^I), R_e^4}, priv(R_k^I)) = Dec(\eta, m_{1,3}, PK) = p_3$
17. $R^2 = p_3$
18. $R_e^5 = r_e^5$
19. $R^2_{pub(R_k^B), R_e^5} = Enc(\eta, p_3, K_B, r_e^5)$
20. $(NSPK, 2, 3, A, R^2_{pub(R_k^B), R_e^5})^I$

In this representation, $m_{i,j}, p_m, r_e^n$ (for $i = 1, 2, j = 1, 2, 3, m = 1, 3, n = 1, 2, 3, 4, 5$) are bitstrings. The strategy of attack described by this algorithm is independent of these bitstrings. This is a natural fact: for example, the exact bitstring corresponding to the encryption of X, $Enc(X)$, usually reveals little about X. If some function F of $Enc(X)$ gives the attacker valuable information about X, the attack strategy would not explicitly use the bitstring corresponding to $Enc(X)$, depending instead on the strings corresponding to $F(Enc(X))$. The variables used to represent bitstrings in this attack are only used to represent the fact that the attacker knows the bitstring corresponding to their respective expressions.

3.2.2 Warinschi's Less Famous Attack to the NSL protocol

The Needham-Schroeder-Löwe mutual authentication protocol (NSL) is a modified version of the Needham-Schroeder protocol presented by Löwe to fix the security issue posed by the man-in-the-middle attack presented above. The only modification is that the name of the

sender is included in the encrypted message sent in the second step of the protocol. In standard notation, it is represented as follows:

$$\begin{aligned} A &\to B : \{(A, R^1)\}_{pub(R_k^B)} \\ B &\to A : \{(B, R^1, R^2)\}_{pub(R_k^A)} \\ A &\to B : \{R^2\}_{pub(R_k^B)} \end{aligned} \tag{3.39}$$

The three rules used to represent this protocol in our framework are a direct adaptation of the rules used to represent the original Needham-Schroeder protocol, and we do not present them here. What we consider an "attack" is also a direct adaptation of the attacks we considered in the NSPK case: complete executions of the protocol in which the attacker impersonates another agent.

In 2000, Warinschi presented an attack to the NSL protocol using a carefully designed encryption scheme based on Elgamal. This encryption scheme still verifies a strong security requirement (indistinguishability under chosen-plaintext attacks), but shares Elgamal's malleability property and has another simple algebraic property. Namely, there exist agent names $B, I \in \mathbf{B}$ (I is the name of the attacker) and an algorithm $W \in \mathbf{Exp}^A$ such that, for all η and all $b, b' \in \mathbf{B}$,

$$W(\eta) \cdot_\eta pair(\eta, B, b, b') = pair(\eta, I, b, b'). \tag{3.40}$$

In terms of expressions, this equality becomes

$$W() \cdot (B, E, E') = (I, E, E'), \tag{3.41}$$

and thus, using the reasoning in 3.1.4, we obtain rules of the form (3.22), (3.23):

$$\frac{E[S'/S] = b) \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{E = b\}} \tag{3.42}$$

$$\frac{(E = b) \in \mathcal{K}}{\mathcal{K} := \mathcal{K} \cup \{E[S'/S] = b\}}, \tag{3.43}$$

where E is any expression and $S = W() \cdot (B, e, e'), S' = (I, e, e')$ (or vice-versa) for some $e, e' \in \mathbf{Exp}^I$.

Using these rules in conjunction with those already described in this section and (3.25), (3.26), we now describe Warinschi's attack to the NSL protocol. We use the same abbreviations which we have used to describe Löwe's attack to NSPK. Note that the letter I for the agent name verifying (3.41) was chosen since we will need the intruder name to verify this property. This means that at least in principle this attack can only be accomplished by an attacker whose agent name I verifies (3.41).

1. $(NSL, 1, 1, A, \{(A, R^1)\}_{pub(R_k^I), R_e^1})$

2. $\{(A, R^1)\}_{pub(R_k^I), R_e^1} = m_{1,1}$
3. $R_k^I = K$
4. $priv(R_k^I) = priv(\eta, K_I) = PK$
5. $Dec(\{(A, R^1)\}_{pub(R_k^I), R_e^1}, priv(R_k^I)) = Dec(\eta, m_{1,1}, PK) = p_1$
6. $(A, R^1) = p_1$
7. $pub(R_k^B) = K_B$
8. $R_e^2 = r_e^2$
9. $\{(A, R^1)\}_{pub(R_k^B), R_e^2} = Enc(\eta, p_1, K_B, r_e^2) = m_{2,1}$
10. $(NSPK, 2, 1, A, \{(A, R^1)\}_{pub(R_k^B), R_e^2})^I$
11. $(NSPK, 2, 2, B, \{(B, R^1, R^2)\}_{pub(R_k^A), R_e^3})$
12. $\{(B, R^1, R^2)\}_{pub(R_k^A), R_e^3} = m_{2,2}$
13. $W() = w$
14. $W() \cdot \{(B, R^1, R^2)\}_{pub(R_k^A), R_e^3} = w \cdot_\eta m_{2,2}$
15. $\{W() \cdot (B, R^1, R^2)\}_{pub(R_k^A), R_e^3} = w \cdot_\eta m_{2,2}$
16. $\{(I, R^1, R^2)\}_{pub(R_k^A), R_e^3} = w \cdot_\eta m_{2,2}$
17. $(NSPK, 1, 2, I, \{(I, R^1, R^2)\}_{pub(R_k^A), R_e^3})^I$
18. $(NSPK, 1, 3, A, \{R^2\}_{pub(R_k^I), R_e^4})$
19. $\{R^2\}_{pub(R_k^I), R_e^4} = m_{1,3}$
20. $Dec(\{R^2\}_{pub(R_k^I), R_e^4}, priv(R_k^I)) = Dec(\eta, m_{1,3}, PK) = p_3$
21. $R^2 = p_3$
22. $R_e^5 = r_e^5$
23. $\{R^2\}_{pub(R_k^B), R_e^5} = Enc(\eta, p_3, K_B, r_e^5)$
24. $(NSPK, 2, 3, A, \{R^2\}_{pub(R_k^B), R_e^5})^I$

This attack is essentially a copy of Löwe's attack to NSPK; the attacker just uses malleability and the specific property of this encryption scheme to obtain a valid encryption of (I, R^1, R^2) with A's public key from (B, R^1, R^2). By contrast, in the NSPK protocol the attacker receives an encryption of (R^1, R^2) with A's public key, and he may simply forward this message to agent A.

3.3 Indistinguishability under Chosen-Plaintext Attacks

Indistinguishability under chosen-plaintext attacks is a security property of asymmetric encryption schemes, commonly referred to as IND-CPA security. Some proofs of the computational security of protocols rely on the assumption that the underlying encryption scheme is IND-CPA secure or verifies some similar property [25]. Abadi and Rogaway also use similar properties in [3]. In chapter 3.2.1 we mentioned that Warinschi presented an attack to the NSL protocol using an encryption scheme with this property.

In this section we translate the notion of IND-CPA security into our framework. Using our framework, we present two equivalent formulations of this concept. One is similar to the standard definition, but simpler; the other is a statement expressed in terms of the cryptographic properties described above.

We conclude this section by pointing out a little inconvenience with this security requirement. Specifically, we consider an IND-CPA secure encryption scheme $(R_k, pub, priv, R_e, Enc, Dec)$, and construct another encryption scheme $(R_k, pub, priv, R_e, Enc', Dec')$. The encryption function Enc' does not reveal anymore about the underlying plaintext than the original encryption function Enc does; however, we will see that the latter encryption scheme is not IND-CPA secure.

3.3.1 Three Characterizations of IND-CPA Security

Before defining IND-CPA security, we need to define the concept of negligible function.

Definition 3.3.1. *Let $f\colon \mathbb{N} \to \mathbb{R}$. We say that f is negligible if, for every $c > 0$, there exists $n \in \mathbb{N}$ such that*

$$k > n \Rightarrow f(k) \leq k^{-c}. \tag{3.44}$$

A common and widely accepted security requirement is that a polynomial-time attacker (*i.e.*, an attacker who, for security parameter η, performs at most η^k computations, where k is some fixed number) has a negligible probability of success. The definition of IND-CPA involves a similar idea.

Definition 3.3.2. *Let $\mathcal{E} = (R_k, pub, priv, R_e, Enc, Dec)$ be an asymmetric encryption scheme. Consider the following experiment:*

- *a security parameter η is fixed*
- *a random key R_k^1 is generated by R_k: $R_k^1 \leftarrow R_k(\eta)$*
- *the corresponding public key $pub(R_k^1)$ is computed and published*
- *a random bit b is chosen using a random generation algorithm R_b; the value of b remains secret*

- *a probabilistic algorithm A returns 0 or 1.*

  A takes as input η and $pub(R_k^1)$ and may use an oracle $\mathcal{O}$ such that

$$\mathcal{O}(m_1, m_2) = Enc(m_b, pub(R_k^1), R_e^i), \tag{3.45}$$

  where $R_e^i \leftarrow R_e(\eta)$ is sampled from R_e each time the oracle is called.

We say that $\mathcal{E}$ verifies the indistinguishability under chosen plaintext attacks property (or that $\mathcal{E}$ is IND-CPA secure) if, for any polynomial-time algorithm A, the function

$$P[A(\eta, pub(R_k^1)) = 1 \mid b = 1] - P[A(\eta, pub(R_k^1)) = 1 \mid b = 0] \tag{3.46}$$

is negligible as a function of η.

Informally, this experience may be described as follows. The attacker queries a certain oracle with pairs of plaintexts valid for security parameter η, but he may only do so a number of times polynomial in η. The oracle either always returns an encryption of the first message, or always returns an encryption of the second message; the goal of the attacker is to find which in polynomial time.

The intuition behind IND-CPA security is that, for a polynomial-time observer, the encryption function reveals nothing about the underlying plaintext: looking at an encryption and two distinct plaintexts, it impossible to say for sure that one of the plaintexts is significantly more likely to be the correct decryption than the other. The next theorem describes this intuition in a rigorous (if a little involved) way and also presents a simpler definition of IND-CPA security.

Theorem 3.3.3. *Let $\mathcal{E} = (R_k, pub, priv, R_e, Enc, Dec)$ be an asymmetric encryption scheme. Then the following are equivalent:*

(1) $\mathcal{E}$ is not IND-CPA secure.

(2) There exist deterministic polynomial-time algorithms $Q, (\cdot)^0, (\cdot)^1, B$ and a probabilistic polynomial-time algorithm R such that

$$\begin{aligned} &P[B(\eta, Q^0, Q^1, \mathcal{O}(Q^0, Q^1), R^1) = 1 \mid b = 1] \\ -&P[B(\eta, Q^0, Q^1, \mathcal{O}(Q^0, Q^1), R^1) = 1 \mid b = 0] \end{aligned} \tag{3.47}$$

  is not negligible as a function of η, where

$$\begin{aligned} Q^0 &\equiv (Q(\eta, k, R^1))^0, \\ Q^1 &\equiv (Q(\eta, k, R^1))^1, \end{aligned} \tag{3.48}$$

  $R^1 \leftarrow R(\eta)$ and k is the public key used in the IND-CPA experiment.

(3) There is an algorithm which compromises the IND-CPA security of $\mathcal{E}$ by performing only one query to the oracle.

Proof. Implication (3) $\Rightarrow$ (1) is trivial.

(2) $\Rightarrow$ (3) is also simple. In the following, we will write $\mathcal{O}_b$ to refer to the oracle $\mathcal{O}$ using bit b in the computations, so that $\mathcal{O}_b(m_0, m_1) = Enc(\eta, m_b, k, R_e(\eta))$ is an encryption of m_b with key k. From (3.47), it is clear that the algorithm described by

1. $r^1 \leftarrow R(\eta)$;
2. compute $Q^0 = (Q(\eta, k, r^1))^0, Q^1 = (Q(\eta, k, r^1))^1$ (recall that the value of k is public);
3. query $\mathcal{O}_b$ with Q^0, Q^1, obtaining $o = Enc(\eta, Q_b, k, R_e(\eta))$;
4. return $B(\eta, Q^0, Q^1, o, r^1)$

compromises the IND-CPA security of $\mathcal{E}$; it is a polynomial-time algorithm (because it is the composition of polynomial-time algorithms) and uses only a query to the oracle, which concludes the demonstration.

The proof of (1) $\Rightarrow$ (2) is a little harder. Let A be an algorithm which compromises the IND-CPA security of $\mathcal{E}$.

Observe that, since A is a polynomial-time algorithm, there is fixed number c such that, for security parameter η, A performs at most η^c queries to the oracle. Thus, we may assume without loss of generality that A always performs exactly η^c queries to the oracle: if this is not the case, we consider an algorithm A' which mimics A until it has an answer (in at most η^c queries), and then performs the remaining queries and returns the result which A would have returned a few queries ago. It is clear that A' still compromises the IND-CPA security of $\mathcal{E}$, and is also a polynomial-time algorithm.

Consider the probabilistic algorithm used by A to generate the two plaintexts with which it queries the oracle for the first time. Its output depends on the security parameter η, the public key k and the randomness involved in the algorithm. Note also that it is a polynomial-time algorithm. Thus, we may represent this algorithm using a random generation algorithm R_1 and a deterministic algorithm Q_1; the output of $Q_1(\eta, k, r_1)$ is the pair of messages used by the attacker to query the oracle for the first time. Let $(\cdot)^0, (\cdot)^1$ be the algorithms to extract the first and second message from Q_1, so that $(Q_1(\eta, k, r_1))^0$ is the first message and $(Q_1(\eta, k, r_1))^1$ is the second.

Without loss of generality, we may assume that the second pair of messages used to query the oracle is generated by a probabilistic algorithm which depends on the security parameter, the public key, the pair of messages used in the first query, the response of the oracle, a randomly generated parameter and possibly some internal computations used to generate the first pair of messages. Thus, we may consider that the second pair of messages used to query the oracle is generated by $Q_2(\eta, k, r_1, q_1^0, q_1^1, o_1, r_2)$, where $q_1^i = (Q_1(\eta, k, r_1))^i, i = 0, 1$.

More generally, for $i = 1, \ldots, \eta^c$, let

$$Q_i(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, o_{i-1}, r_i) \tag{3.49}$$

be the algorithm which generates the pair of messages used in the i-th query to the oracle. In (3.49), R_j is the random generation algorithm which represents the randomness involved in computing Q_j, $q_j^0 = (Q_1(\eta, k, r_1))^0, q_j^1 = (Q_1(\eta, k, r_1))^1$ are the messages used in the j-th query to the oracle and o_j is the response of the oracle to the j-th query. As before, we used $(\cdot)^0, (\cdot)^1$ as the algorithms which extract, respectively, the first and second message from each Q_i. Each Q_i is computable in polynomial-time: in the worst case, Q_i needs to emulate the executions of $Q_1, \ldots, R_{i-1}$ using random data $r_1, \ldots, r_{i-1}$ (since the output may depend on computations performed by these algorithms). But since $i < \eta^c$, this involves computing at most a polynomial number of algorithms computable in polynomial time, and thus may be done in polynomial time.

When all the queries have been made, there is another algorithm, say B_F, which receives as input the security parameter, the public key, the queries, the answers and the internal state of the program (represented by the randomly generated data $r_1, \ldots, r_\eta^c$), and returns 0 or 1. Thus, the final response of A is given by

$$B_F(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{\eta^c}, q_{\eta^c}^0, q_{\eta^c}^1, o_{\eta^c}, r_{\eta^c+1}) \tag{3.50}$$

(recall our assumption that A always performs exactly η^c queries).

Summarizing, A may be represented as follows:

1. Input: security parameter η, public key k
2. For $i = 1, \ldots, \eta^c$:
   (a) $r_i \leftarrow R_i$
   (b) $q_i^k \leftarrow (Q_i(\eta, k, r_1, q_1^0, q_1^1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, r_i))^k, k = 0, 1$
   (c) $o_i \leftarrow \mathcal{O}_b(q_i^0, q_i^1)$
3. Return $B_F(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{\eta^c}, q_{\eta^c}^0, q_{\eta^c}^1, o_{\eta^c}, r_{\eta^c+1})$.

Observe that we may assume, without loss of generality, that the algorithms Q_i and B_F are total. This is because, since they are computable in polynomial time, there exists a c' such that, for security parameter η, Q_i, B_F performs at most $\eta^{c'}$ operations. Thus, we may define algorithms Q_i', B_F' which emulate Q_i, B_F for at most $\eta^{c'}$ operations. If Q_i, B_F return an answer in this time, Q_i', B_F' return the same answer; otherwise they return some predetermined value. Q_i', B_F' are still computable in polynomial time, and it is clear that replacing Q_i, B_F by Q_i', B_F' in the algorithm A does not affect its output.

Now suppose that, instead of using the same bit b in all queries, the oracle $\mathcal{O}$ uses the bit b_i in the i-th query (in other words, in the i-th query A is querying $\mathcal{O}_{b_i}$). Let $\delta_{i \geq j} = 1$ if

$i \geq j$ and 0 otherwise. Note that

$$P[A(\eta,k)=1 \mid b_i=1] - P[A(\eta,k)=1 \mid b_i=0] = \\ \sum_{j=1}^{\eta^c}(P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j}] - P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j+1}]), \tag{3.51}$$

because the sum on the right-hand side is telescopic. The left-hand side is non-negligible by hypothesis, and thus so is the right hand side. We conclude that there is some c' such that, for all η,

$$\sum_{j=1}^{\eta^c}(P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j}] - P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j+1}]) > \frac{1}{\eta^{c'}}. \tag{3.52}$$

The assumption that Q_i, B_F return an answer in polynomial time for every possible input is necessary here, since we are running these algorithms with inputs that would be impossible in the original setting. For example, in the original setting it would be impossible to have o_i as an encryption of q_i^0 and o_{i+1} as an encryption of q_{i+1}^1, since the oracle would either always encrypt the first message or always encrypt the second one.

Querying the oracle $\mathcal{O}_b$ with q_0, q_1 is the same as computing $Enc(\eta, q_b, k, R_e(\eta))$ - that is, the outputs of the two computations have the same distribution. In particular, if in some step of the algorithm A, instead of querying the oracle $\mathcal{O}_b$ with plaintexts q_i^0, q_i^1, we encrypt q_i^b with public key k and random data r_i, the probability distribution of the output does not change.

Consider the following modified version A' of the algorithm A. For each η, A' receives an additional argument $(t_i)_{i=1,\ldots,\eta^c}$. Each t_i is either 0, 1 or *oracle*. A' may be described as follows:

1. Input: security parameter η, public key k, sequence $(t_i)_{i=1,\ldots,\eta^c}$
2. For $i = 1, \ldots, \eta^c$:
   (a) $r_i \leftarrow R_i$
   (b) $(q_i^0, q_i^1) \leftarrow Q_i(\eta, k, r_1, q_1^0, q_1^1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, r_i)$
   (c) $o_i \leftarrow \begin{cases} \mathcal{O}_b(q_i^0, q_i^1) & \text{if } t_i = oracle \\ Enc(\eta, q_i^0, k, R_e^i) & \text{if } t_i = 0 \\ Enc(\eta, q_i^1, k, R_e^i) & \text{if } t_i = 1 \end{cases}$
3. Return $B_F(\eta, k, R_1, q_1^0, q_1^1, o_1, \ldots, R_{\eta^c}, q_{\eta^c}^0, q_{\eta^c}^1, o_{\eta^c}, R_{\eta^c+1})$.

Fix $\eta \in \mathbb{N}$, and let $T^j = (T_i^j)_{i=1,\ldots,\eta^c}$, where

$$T_i^j = \begin{cases} 0 & \text{if } i < j \\ oracle & \text{if } i = j \\ 1 & \text{if } i > j \end{cases} \tag{3.53}$$

Following the reasoning above, it is easy to see that:

$$P[A'(\eta,k,T^j)=1]=\begin{cases} P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j+1}] & \text{if } b=0 \\ P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j}] & \text{if } b=1 \end{cases} \tag{3.54}$$

Thus,

$$\begin{gathered} P[A'(\eta,k,T^j)=1 \mid b=1]-P[A'(\eta,k,T^j)=1 \mid b=0]= \\ P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j}]-P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j+1}]. \end{gathered} \tag{3.55}$$

Suppose now that A'' is an algorithm which chooses $j=1,\ldots,\eta^c$ randomly (with uniform distribution) and then executes $A'(\eta,k,T^j)$. We obtain:

$$\begin{gathered} P[A''(\eta,k)=1 \mid b=1]-P[A''(\eta,k)=1 \mid b=0]= \\ \sum_{l=1}^{\eta^c} P[j=l]\cdot(P[A'(\eta,k,T^l)=1 \mid b=1]-P[A'(\eta,k,T^l)=1 \mid b=0])= \\ \frac{1}{\eta^c}\sum_{j=1}^{\eta^c}(P[A'(\eta,k,T^j)=1 \mid b=1]-P[A'(\eta,k,T^j)=1 \mid b=0])= \\ \frac{1}{\eta^c}\sum_{j=1}^{\eta^c}(P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j}]-P[A(\eta,k)=1 \mid b_i=\delta_{i\geq j+1}])= \\ \frac{1}{\eta^c}(P[A(\eta,k)=1 \mid b=1]-P[A(\eta,k)=1 \mid b=0]). \end{gathered} \tag{3.56}$$

Thus, by (3.51) and the equalities above, we conclude that

$$P[A''(\eta,k)=1 \mid b=1]-P[A''(\eta,k)=1 \mid b=0] \tag{3.57}$$

is non-negligible.

It is clear that A'' is a polynomial-time algorithm: it involves η^c steps, each of which consists of encrypting a message (or a query to the oracle) and running an algorithm Q_i. We have already seen that the latter may be done in polynomial time, and it is clear that the same holds for the former. Furthermore, A'' involves only one query to the oracle, and compromises the IND-CPA security of $\mathcal{E}$. To complete the proof we need only see that A'' may be written in terms of algorithms $Q,(\cdot)^0,(\cdot)^1,B,R$. This may be done by defining these algorithms as follows:

- R:
  1. Generate a random $j\in\{1,\ldots,\eta^c\}$ (with uniform probability distribution)
  2. For $i=1,\ldots,\eta^c$, $r_i \leftarrow R_i(\eta)$

3. For $i = 1, \ldots, \eta^c$, $R_e^i \leftarrow R_e(\eta)$
4. Return an encoding of $(j, r_1, \ldots, r_{\eta^c}, R_e^1, \ldots, R_e^{\eta^c})$

- Q:

1. Decode R, obtaining $(j, r_1, \ldots, r_{\eta^c}, R_e^1, \ldots, R_e^{\eta^c})$
2. For $i = 1, \ldots, j-1$:

$$q_i = Q_i(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, o_{i-1}, r_i);$$

$$o_i = Enc(\eta, q_i^0, k, R_e^i)$$

3. Return q_j

- B:

1. Decode R, obtaining $(j, r_1, \ldots, r_{\eta^c}, R_e^1, \ldots, R_e^{\eta^c})$
2. For $i = 1, \ldots, j-1$:

$$q_i = Q_i(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, o_{i-1}, r_i)$$

$$o_i = Enc(\eta, q_i^0, k, R_e^i)$$

3. $q_j = (Q^0, Q^1)$
4. $o_j = \mathcal{O}_b(q_{j-1}^0, q_{j-1}^1)$
5. For $i = j+1, \ldots, \eta^c$:

$$q_i = Q_i(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{i-1}, q_{i-1}^0, q_{i-1}^1, o_{i-q}, r_i)$$

$$o_i = Enc(\eta, q_i^0, k, R_e^i)$$

6. Return
$$B_F(\eta, k, r_1, q_1^0, q_1^1, o_1, \ldots, r_{\eta^c}, q_{\eta^c}^0, q_{\eta^c}^1, o_{\eta^c}, r_{\eta^c+1}).$$

- $(\cdot)^0, (\cdot)^1$ function in the way described above.

It is now straigthforward to check that computing

$$B(\eta, (Q(\eta, k, R^1))^0, (Q(\eta, k, R^1))^1, \mathcal{O}(Q^0, Q^1), R^1)$$

(where $R^1 \leftarrow R$) is the same as running algorithm A'', which concludes the demonstration. □

Notice that it was not necessary to include Q_0, Q_1 as arguments in B. We prefer to do it for intuitive reasons. The attacker presented should be interpreted as an algorithm which carefully determines messages Q^0, Q^1 and queries the oracle with them. The encryption

returned by the oracle in response to these queries is enough for B to break IND-CPA security, and the decision of returning 0 or 1 depends on the queries made to the oracle as much as on the response of the oracle.

This example is important because it shows us how to represent a standard notion of security in terms of the properties which we consider in our framework. Furthermore, it is simple to see that the IND-CPA experiment may itself be represented in our framework as a protocol would be.

3.3.2 An Unsatisfactory Fact About IND-CPA Security

As mentioned before, IND-CPA intuitively means that the attacker is not able to obtain any relevant information about the plaintext by looking at an encryption of it. However, there is a somewhat unsatisfactory fact about this definition. We will show how to alter an IND-CPA encryption scheme so that the new encryption function reveals the same information as the original one, but the encryption scheme so obtained is no longer IND-CPA secure.

More precisely, for a given asymmetric encryption scheme $\mathcal{E}$, we will construct an essentially similar asymmetric encryption scheme $\mathcal{E}_{IND}$. $\mathcal{E}_{IND}$, however, is not IND-CPA secure even if $\mathcal{E}$ is.

Definition 3.3.4. *Let $\mathcal{E} = (R_k, pub, priv, R_e, Enc, Dec)$ be an asymmetric encryption scheme. We define*

$$\mathcal{E}_{IND} = (R'_k, pub', priv', R'_e, Enc', Dec') \tag{3.58}$$

as the encryption scheme such that:

- $R'_k = R_k, pub' = pub, priv' = priv, R'_e = R_e$
- $$Enc' \colon \bigcup_{\eta \in \mathbb{N}} (\{\eta\} \times \mathbf{Plain}'(\eta) \times \mathbf{PubKeys}(\eta) \times \mathbf{Random}(\eta)) \to \mathbf{B}, \tag{3.59}$$

  where $\mathbf{Plain}'(\eta) = \{0,1\} \times \mathbf{Plain}(\eta)$ and $\mathbf{Plain}(\eta)$ is the set of valid plaintexts for the encryption scheme $\mathcal{E}$ and security parameter η, is computed as follows:

  $$Enc'(\eta, p' = (b, p), pubK, r) = (b, Enc(\eta, p, pubK, r)). \tag{3.60}$$

  In this equation, $p \in \mathbf{Plain}(\eta)$ and $b \in \{0,1\}$, so that $p' = (b,p) \in \mathbf{Plain}'(\eta)$.
- *Dec' is computed as follows:*

  $$Dec'(\eta, c' = (b, c), privK) = (b, Dec(\eta, c, privK)) \tag{3.61}$$

  (note that, from the definition of Enc', we have that if the set of valid ciphertexts for $\mathcal{E}$ and security parameter η is $\mathbf{Cipher}(\eta)$, then the set of valid ciphertexts for $\mathcal{E}_{IND}$

is $\mathbf{Cipher}'(\eta) = \{0,1\} \times \mathbf{Cipher}(\eta)$*).*

Intuitively, each valid plaintext in $\mathcal{E}_{IND}$ is a valid plaintext of $\mathcal{E}$ with one bit prepended. The encryption function keeps this bit and encrypts the rest of the message as $\mathcal{E}$ would have.

It is clear that $\mathcal{E}_{IND}$ is not IND-CPA secure: if one generates a random $p \in \mathbf{Plain}(\eta)$ and queries the oracle with $(0,p),(1,p)$, the first bit of the response of the oracle will be 0 if it encrypts the first message and 1 otherwise. This is a simple algorithm which gives an adversary probability 1 of correctly guessing the bit used by the oracle. We have thus obtained:

Theorem 3.3.5. *$\mathcal{E}_{IND}$ is not IND-CPA secure.*

However, it is not any easier for an attacker to break $\mathcal{E}_{IND}$ than to break $\mathcal{E}$. In fact, let $\mathcal{E} = (R_k, pub, priv, R_e, Enc, Dec)$ be any asymmetric encryption scheme and, for each $\eta \in \mathbb{N}$, let X_η be a random variable taking values in the set $\mathbf{Plain}(\eta)$ of valid plaintexts for security parameter η. Fix some random variable b taking values in $\{0,1\}$, and define $X'_\eta = (b, X_\eta)$. X'_η is a random variable taking values in the set $\mathbf{Plain}'(\eta)$ of valid plaintexts for security parameter η for the encryption scheme $\mathcal{E}_{IND}$. Consider an attacker who looks at an encrypted message $Enc(X'_\eta, k, R^1_e)$ and attemps to find X'_η by computing $c'(\eta, k, Enc(X'_\eta, k, R^1_e), R^1_c)$ (we assume that the attacker knows the public key being used). Random data R^1_c is used to represent the randomness involved in the computations.

Whatever the algorithm c' is, and whatever its probability of sucess, the algorithm c computed by

$$c(\eta, k, Enc(X_\eta, k, R^1_e), R^1_c) = c'(\eta, k, (0, Enc(X_\eta, k, R^1_e)), R^1_c) \tag{3.62}$$

is also a polynomial-time algorithm, and

$$P[c(\eta, k, Enc(X_\eta, k, R^1_e), R^1_c) = X_\eta] \geq P[c'(\eta, k, Enc'(X_\eta, k, R^1_e), R^1_c) = X'_\eta] \tag{3.63}$$

(since $c'(\eta, k, Enc'(X_\eta, k, R^1_e), R^1_c) = X'_\eta$ implies $c(\eta, k, Enc(X_\eta, k, R^1_e), R^1_c) = X_\eta$). This means that if the attacker "breaks" $\mathcal{E}_{IND}$ (by looking at an encryption and guessing the underlying message) then it may break $\mathcal{E}$ with at least the same probability. The same is true if we allow the algorithms c', c to obtain information from an arbitrary number (more than one) of encryption of plaintexts.

It is clear that the encryption function Enc' does reveal partial information about the plaintext, while the original function Enc may not. However, the set of valid plaintexts is greater in $\mathcal{E}_{IND}$. The partial information revealed by Enc' allows the attacker to overcome this additional difficulty, but otherwise does not reveal any information that is not also revealed by Enc. The example above, in particular, shows us that the partial information revealed by Enc' is not sufficient to guess the plaintext more efficiently than it would be possible with the information revealed by Enc.

Chapter 4

Probabilistic Reasoning Using Cryptographic Properties

In this section we discuss how to estimate a probability distribution of the random variables represented by expressions based on the cryptographic properties known by the attacker. This is useful for estimating the probability of success of an attacker (by using the attacker's knowledge to describe known properties of cryptographic primitives used in the protocol). Furthermore, note that we may express general cryptographic properties, which may or may not apply to a given cryptographic primitive. As such, estimating this probability distribution for given cryptographic properties (instead of using cryptographic properties verified by predetermined cryptographic primitives) allows us to study whether or not those properties compromise the security of a given protocol.

We describe a method for obtaining an estimation of this distribution and present some properties which make it a reasonable estimation. Not surprisingly, we see that the problem of calculating probabilities of this estimated distribution is at least NP-hard in the number of cryptographic properties known by the attacker. Note that this does not necessarily mean that it is NP-hard in the security parameter.

4.1 Estimating a Distribution

Recall from section 3.1 that we consider an attacker with access to a function

$$p\colon \mathbb{N} \times ((\mathbf{E} \times \mathbf{B})^*)^2 \to [0,1] \cup \{\perp\}$$

such that, if

$$p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho \in [0,1],$$

then

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*] = \rho.$$

We want to determine a function

$$p^{est}\colon \mathbb{N} \times ((\mathbf{E} \times \mathbf{B})^*)^2 \to [0,1] \cup \{\bot\}$$

such that, for any $\eta \in \mathbb{N}$, $E^1, \ldots, E^n$, $E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{E}^A$, and $b_1, \ldots, b_n$, $b_1^*, \ldots, b_{n^*}^* \in \mathbf{B}$,

$$p_\eta^{est}[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*]$$

(we will use a notation analogous to the one described for the function p in (3.11), (3.12)) is a "reasonable" estimation of

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*]$$

given the knowledge provided to the attacker by the oracle p.

Before we proceed let us fix some notation. Note that an attacker with limited computational power (*i.e.*, an attacker who, for a given security parameter η, may only perform a certain limited number $f(\eta)$ of operations) may only interpret and obtain information from expressions with at most $f(\eta)$ subexpressions (for security parameter η), since he cannot parse larger ones. As such, when modelling an adversary with limited (but arbitrary) computational power for a fixed security parameter η, the set of the expressions which we need to consider is finite. Let us denote that set by $\mathbf{Exp}_\eta^A \subseteq \mathbf{Exp}^A$. Observe that we need to assume a finite set of expressions of the form R^i for each algorithm $R \in \mathcal{R}^A$; we may require, for example, that $i < f(\eta)$.

For similar reasons, we may assume that the set of bitstrings which an expression in $\mathbf{Exp}_\eta$ may possibly represent is also a finite set $\mathbf{B}_\eta \subseteq \mathbf{B}$. For the remainder of this section, we will interpret expressions in $\mathbf{Exp}_\eta^A$ as random variables taking values in $\mathbf{B}_\eta$. We will denote the set of possible values for a random variable E by $\mathcal{A}(E)$.

For each η, we define the set

$$\mathcal{U}_\eta = \prod_{E \in \mathbf{Exp}_\eta^A} \mathbf{B}_\eta. \tag{4.1}$$

Intuitively, an element of $\mathcal{U}_\eta$ associates each expression in $\mathbf{Exp}_\eta$ to a bitstring in $\mathbf{B}_\eta$. Thus, if $\mathbf{Exp}_\eta = \{E^1, \ldots, E^N\}$, an element of $\mathcal{U}_\eta$ is an N-uple of bitstrings, each corresponding to an expression. As such, if $\mathcal{D}$ is a probability distribution on $\mathcal{U}_\eta$, we write $(E^1, \ldots, E^N) \leftarrow \mathcal{D}$ to denote the sampling of an element of $\mathcal{U}_\eta$ from $\mathcal{D}_\eta$. Similarly, we write $E^1, \ldots, E^n \leftarrow \mathcal{D}_\eta$ when $E^1, \ldots, E^n \subseteq \mathbf{Exp}_\eta^A$. If $x \in \mathcal{U}_\eta$ and $E \in \mathbf{Exp}_\eta$, we will also write x_E to denote the bitstring associated to the expression E by the element x of $\mathcal{U}_\eta$.

If $E^1, \ldots, E^n$, $E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{Exp}_\eta$, $b_1, \ldots, b_n$, $b_1^*, \ldots, b_{n^*}^* \in \mathbf{B}_\eta$ and $\rho \in [0,1]$, we will say that the cryptographic property

$$P[E_\eta^1 = b_1, \ldots, E_\eta^n = b_n \mid E_\eta^{*,1} = b_1^*, \ldots, E_\eta^{*,n^*} = b_{n^*}^*] = \rho, \tag{4.2}$$

is verified by a probability distribution $\mathcal{D}$ in $\mathcal{U}_\eta$ if

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho \tag{4.3}$$

when $(E^1, \ldots, E^n, E^{*,1}, \ldots, E^{*,n^*}) \leftarrow \mathcal{D}$.

We will denote by $\mathcal{D}_\eta$ the probability distribution on $\mathcal{U}_\eta$ defined by

$$\begin{gathered} P[E^1 = b_1, \ldots, E^N = b_N \mid (E^1, \ldots, E^N) \leftarrow \mathcal{D}_\eta] = \\ P[E_\eta^1 = b_1, \ldots, E_\eta^N = b_N], \end{gathered} \tag{4.4}$$

where $\mathbf{Exp}_\eta = \{E^1, \ldots, E^N\}$ and the equality above is valid for any values of $b_1, \ldots, b_N \in \mathbf{B}_\eta$. Intuitively, $\mathcal{D}_{eta}$ represents the "real" probability distribution in $\mathcal{U}_\eta$. Note that in light of the previous discussion we assume here that $\mathcal{A}(E_\eta) \subseteq \mathbf{B}_\eta$ for $E \in \mathbf{Exp}_\eta$.

Exactly what is a "reasonable" estimation is difficult to define, and such problems have been subject of some research, namely on inductive logic [16]. We require that this estimation verifies at least the following two properties:

- if

$$p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho \in [0, 1], \tag{4.5}$$

  then

$$p_\eta^{est}[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho; \tag{4.6}$$

- for any $E^1, \ldots, E^n$, $E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{E}_\eta$, $b_1^*, \ldots, b_{n^*}^* \in \mathbf{B}_\eta$,

$$\begin{gathered} \sum_{b_1, \ldots, b_n \in \mathbf{B}} p_\eta^{est}[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = 1, \\ p_\eta^{est}[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] \geq 0. \end{gathered} \tag{4.7}$$

The first of these properties states that p^{est} agrees with p whenever p returns a numeric value. The second one states that p_η^{est} determines a probability distribution in $\mathcal{U}_\eta$. In fact, let $\mathbf{Exp}_\eta = \{E^1, \ldots, E^N\}$. It is clear that a function p^{est} which verifies (4.7) determines a probability distribution $\mathcal{D}^{est}$ on the set

$$\mathcal{U}_\eta = \prod_{i=1}^{N} \mathbf{B}_\eta, \tag{4.8}$$

by setting

$$P[E^1 = b_1, \ldots, E^N = b_N] = p_\eta^{est}[E^1 = b_1, \ldots, E^N = b_N]. \tag{4.9}$$

Conversely, a distribution of probability $\mathcal{D}$ on $\mathcal{U}_\eta$ determines a function p_η^{est}, by defining

$$\begin{gathered} p_\eta^{est}[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \\ = P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] \end{gathered} \tag{4.10}$$

where $(E^1, \ldots, E^N) \leftarrow \mathcal{D}$, $E^1, \ldots, E^n$, $E^{*,1}, \ldots, E^{*,n^*} \in \mathbf{Exp}_\eta$ are any expressions, and $P[E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] > 0$.

We want to estimate these probabilities based on the known values of the function p. This allows us to estimate the probability of success of an attacker given what he knows. This comprises both cryptanalytical knowledge and the knowledge which the attacker is able to obtain about the messages published during the execution of the protocol.

Our goal is to study how cryptographic properties influence the probability of success of the attacker. Thus, we want to find an estimation which (informally speaking) includes all the knowledge of the attacker but otherwise keeps the problem of finding an attack as hard as possible.

In the following we explore a natural approach to accomplish such a goal and discuss some properties which make it a reasonable "solution". We recall the definition of Shannon entropy $H(X)$ for a random variable X [24].

Definition 4.1.1. *Let X. The Shannon entropy of X is*

$$H(X) = -\sum_{x \in \mathcal{A}(X)} P[X = x] * \log P[X = x], \tag{4.11}$$

where we stipulate that $0 * \log 0 = 0$.

Shannon entropy is widely used in information theory as the standard measure for uncertainty about the outcome of a random experiment. We recall some of the properties which make it suited to this purpose.

Theorem 4.1.2. *Let X be a random variable such that the set $\mathcal{A}(X)$ is finite. The Shannon entropy function H verifies the following properties:*

(i) $0 \leq H(X) \leq \log|\mathcal{A}(X)|$;

(ii) $H(X) = 0$ *iff* $P[X = x] = 1$ *for some* $x \in \mathcal{X}$;

(iii) $H(X) = \log|\mathcal{A}(X)|$ *iff X has uniform probability distribution (*i.e., $P[X = x] = P[X = y]$ *for all* $x, y \in \mathcal{A}(X)$*).*

Property (ii) states that the entropy (or uncertainty) of the outcome of a random event which takes some value with probability 1 is 0. This is a desirable property, since it is intuitive that in this case there is no uncertainty. Similarly, properties $(i), (iii)$ assure us that the entropy is greater when the probability distribution is uniform.

Shannon entropy also has good properties when measuring the uncertainty of a random variable X in presence of information about another random variable Y, whose outcome may or may not be related to the outcome of X. More precisely, we define the Shannon entropy of a random variable X conditioned on Y as follows.

Definition 4.1.3. *Let X, Y be random variables such that the sets $\mathcal{A}(X), \mathcal{A}(Y)$ are finite. The Shannon entropy of X conditioned on Y is given by*

$$H(X \mid Y) = -\sum_{y \in \mathcal{A}(Y)} \sum_{x \in \mathcal{A}(X)} P[Y = y] \cdot (P[X = x \mid Y = y] \log P[X = x \mid Y = y]). \quad (4.12)$$

$H(X \mid Y)$ may be viewed as the expected value of $H(X \mid Y = y)$ when y is randomly sampled from a random variable with the same distribution as Y: each possible value y of Y corresponds to some uncertainty $H(X \mid Y = y)$, and $H(X \mid Y)$ is the average value of that uncertainty when y is randomly sampled. The following theorem presents some important properties of this definition.

Theorem 4.1.4. *Let X, Y be random variables such that the sets $\mathcal{A}(X), \mathcal{A}(Y)$ are finite. The Shannon entropy of X conditioned on Y verifies:*

1. *$H(X \mid Y) \leq H(X)$, and $H(X \mid Y) = H(X)$ iff $X \perp Y$ (*i.e.*, X and Y are independent);*
2. *$H(X, Y) = H(Y) + H(X \mid Y)$,*

where (X, Y) is the random variable obtained by pairing the outcomes of X and Y.

In particular, $H(X, Y) \leq H(X) + H(Y)$, and the equality holds iff $X \perp Y$.

The first property states that knowing the value of Y cannot increase the uncertainty of X. The second property is also a good property for an uncertainty measure: it says that the uncertainty of (X, Y) is the uncertainty of Y plus the uncertainty of X when we already know Y. $H(X, Y)$ is smaller if the uncertainty of X given Y is small, *i.e.*, if Y reveals a lot of information about X. The maximum value of $H(X, Y)$ is achieved if $X \perp Y$, in which case Y reveals no information about X and the uncertainty of (X, Y) is just the uncertainty of X plus the uncertainty of Y.

The method for estimating probability distributions of random variables represented by expressions is based in this concept. We have already mentioned that we want our estimated distribution to use all the knowledge of the attacker, but otherwise be as hard to attack as possible. Thus, we will use a distribution with as great an entropy as possible while still verifying all the cryptographic properties prescribed. A precise definition follows.

Definition 4.1.5. *Given a problem of the previously described form for the set $\mathbf{Exp}_\eta = \{E^1, \ldots, E^N\}$ and the function p, we say that $\mathcal{D}_\eta^H$ is a distribution of maximum entropy in $\mathcal{U}_\eta = \prod_{i=1}^N \mathbf{B}_\eta$ if the following conditions are verified:*

- *if*

$$p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho \neq \perp, \quad (4.13)$$

and $(E^1, \ldots, E^n, E^{*,1}, \ldots, E^{*,n^*}) \leftarrow \mathcal{D}_\eta^H$, then

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho. \quad (4.14)$$

- *if $\mathcal{D}^*$ is another distribution which respects the first property, and X, X^* are random variables taking values in $\mathcal{U}_\eta$ with distribution $\mathcal{D}_\eta^H, \mathcal{D}^*$, then*

$$H(X) \geq H(X^*). \tag{4.15}$$

The set $\mathcal{U}_\eta$ is a finite cartesian product of finite sets. Thus, $\mathcal{U}_\eta$ is finite, and the entropy $H(X)$ is well defined for $X = (E^1, \ldots, E^N) \leftarrow \mathcal{D}_\eta^H$.

We may assume that the set of cryptographic properties known by the attacker for a fixed security parameter η is finite, by requiring that p only returns an answer when the input is in the set

$$\{\eta\} \times ((\mathbf{Exp}_\eta \times \mathbf{B}_\eta)^*)^2. \tag{4.16}$$

We will denote the (finite) set of cryptographic properties known by the attacker for security parameter η by $\mathbf{R}_\eta$.

If all outputs of the function p are correct (*i.e.*, $\mathcal{D}_\eta$ verifies all cryptographic properties computed by p), then there is at least one distribution (the "real" one) which verifies all properties in $\mathbf{R}_\eta$. If the set of distributions which verify all properties in $\mathbf{R}_\eta$ is non-empty, then the problem of finding the distribution of maximum entropy is essentially maximizing

$$\sum_{\substack{(E,b)\in \\ \mathbf{Exp}_\eta \times \mathbf{B}_\eta}} x(E,b) \log x(E,b) \tag{4.17}$$

restricted to the set

$$\{x = \prod_{\substack{(E,b)\in \\ \mathbf{Exp}_\eta \times \mathbf{B}_\eta}} x(E,b) \colon \sum_{\substack{(E,b)\in \\ \mathbf{Exp}_\eta \times \mathbf{B}_\eta}} x(E,b) = 1, x(E,b) \geq 0 \text{ for all } E, b\}. \tag{4.18}$$

This is a continuous function in a non-empty, compact set. Thus, there is always a maximum, and at least one distribution of maximum entropy exists.

We will now present some important properties of $\mathcal{D}_\eta^H$ which help to justify why we choose it as a reasonable estimation. We will use the following definition.

Definition 4.1.6. *Let $r \in \mathbf{R}_\eta$ be the cryptographic property*

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho, \tag{4.19}$$

and let

$$\begin{aligned} &\mathcal{U}_\eta \supseteq A = \{x \in \mathcal{U}_\eta \colon x_{E^1} = b_1, \ldots, x_{E^N} = b_N\}, \\ &\mathcal{U}_\eta \supseteq A^* = \{x \in \mathcal{U}_\eta \colon x_{E^{*,1}} = b_1^*, \ldots, x_{E^{*,n^*}} = b_{n^*}^*\}. \end{aligned} \tag{4.20}$$

We will say that the set $\{P_r^1, P_r^2, P_r^3\}$ is the partition (of $\mathcal{U}_\eta$) induced by r, where

- $P_r^1 = A^* \setminus A$;

- $P_r^2 = A \cap A^*$;
- $P_r^3 = \mathcal{U}_\eta \setminus A^*$.

Observe that $\bigcup_{i=1}^3 P_r^i = \mathcal{U}_\eta$ *and* $P_r^i \cap P_r^j = \emptyset$ *for any* $i \neq j$.

If $\mathbf{R}_\eta = \{r_1, \ldots, r_n\}$ *is a finite set of cryptographic properties, we say that*

$$\bigcup_{i_1=1}^{3} \ldots \bigcup_{i_n=1}^{3} \{P_{r_1}^{i_1} \cap \ldots \cap P_{r_n}^{i_n}\} \tag{4.21}$$

is the partition (of $\mathcal{U}_\eta$*) induced by* $\mathbf{R}_\eta$.

The following theorem illustrates the importance of this definition. For each $r \in \mathbf{R}_\eta$ and each $x \in \mathcal{U}_\eta$, we will write $i(r, x)$ for the number $i \in \{1, 2, 3\}$ such that $x \in P_r^{i(r,x)}$.

Theorem 4.1.7. *Fix a security parameter* η. *Let* $\mathbf{R}_\eta = \{r_1, \ldots, r_N\}$ *be the set of cryptographic properties known by the attacker for security parameter* η, *and consider the partition of* $\mathcal{U}_\eta$ *induced by* $\mathbf{R}_\eta$.

Suppose that there is a distribution of maximum entropy $\mathcal{D}_\eta^H$ *according to definition 4.1.5 for* $\mathbf{R}_\eta$, *and fix* $(i_1, \ldots, i_N) \in \{1, 2, 3\}^N$. *Then, for* $X \leftarrow \mathcal{D}^H$, *we have*

$$x, y \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N} \Rightarrow P[X = x] = P[X = y]. \tag{4.22}$$

Proof. We will consider an arbitrary distribution $\mathcal{D}$ which verifies all properties in $\mathbf{R}_\eta$ and obtain another distribution $\mathcal{D}'$. We will then show that properties in $\mathbf{R}_\eta$ still hold for $\mathcal{D}'$, which also verifies (4.22) and has greater entropy than $\mathcal{D}$.

We define the distribution $\mathcal{D}'$ by

$$P[X' = x] = \frac{P[X \in P_{r_1}^{i(r_1,x)} \cap \ldots \cap P_{r_N}^{i(r_N,x)}]}{|P_{r_1}^{i(r_1,x)} \cap \ldots \cap P_{r_N}^{i(r_N,x)}|}, \tag{4.23}$$

where $X' \leftarrow \mathcal{D}'$.

To see that $\mathcal{D}'$ verifies all properties in $\mathbf{R}_\eta$, fix j and suppose that

$$r_j = (p_\eta[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho). \tag{4.24}$$

Now, if $X' \leftarrow \mathcal{U}_\eta$, we have

$$\begin{array}{l} P[X'_{E^1} = b_1, \ldots, X'_{E^n} = b_n \mid X'_{E^{*,1}} = b_1^*, \ldots, X'_{E^{*,n^*}} = b_{n^*}^*] = \\ P[X' \in P_{r_j}^2 \mid X' \in P_{r_j}^1 \cup P_{r_j}^2] = \\ \quad \dfrac{P[X' \in P_{r_j}^2]}{P[X' \in P_{r_j}^1 \cup P_{r_j}^2]}. \end{array} \tag{4.25}$$

For each sequence $(i_1, \ldots, i_N) \in \{1,2,3\}^N$, it is clear that

$$\begin{aligned}
&P[X' \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}] = \\
&\sum_{x \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}} P[X' = x] = \\
&\sum_{x \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}} \frac{P[X \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}]}{|P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}|} = \\
&|P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}| \cdot \frac{P[X \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}]}{|P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}|} = \\
&P[X \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}].
\end{aligned} \tag{4.26}$$

From this, by setting $i_j = 2$, we conclude that

$$\begin{aligned}
&P[X' \in P^2_{r_j}] = \\
&\sum_{i_1, \ldots, \hat{i_j}, \ldots, i_N} P[X' \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}] = \\
&\sum_{i_1, \ldots, \hat{i_j}, \ldots, i_N} P[X \in P^{i_1}_{r_1} \cap \ldots \cap P^{i_N}_{r_N}] = \\
&P[X \in P^2_{r_j}].
\end{aligned} \tag{4.27}$$

A similar reasoning shows that $P[X' \in P^1_{r_j} \cup P^2_{r_j}] = P[X \in P^1_{r_j} \cup P^2_{r_j}]$. Thus,

$$\begin{aligned}
&P[X'_{E^1} = b_1, \ldots, X'_{E^n} = b_n \mid X'_{E^{*,1}} = b^*_1, \ldots, X'_{E^{*,n^*}} = b^*_{n^*}] = \\
&\frac{P[X' \in P^2_{r_j}]}{P[X' \in P^1_{r_j} \cup P^2_{r_j}]} = \\
&\frac{P[X \in P^2_{r_j}]}{P[X \in P^1_{r_j} \cup P^2_{r_j}]} = \\
&P[X_{E^1} = b_1, \ldots, X_{E^n} = b_n \mid X_{E^{*,1}} = b^*_1, \ldots, X_{E^{*,n^*}} = b^*_{n^*}].
\end{aligned} \tag{4.28}$$

Since we know that $\mathcal{D}$ verifies properties $\mathbf{R}_\eta$, we conclude that $\mathcal{D}'$ does too.

We now check that $H(X') \geq H(X)$. From Jensen's inequality for convex functions, we know that

$$\sum_{i=1}^{n} x_i = k \Rightarrow -\sum_{i=1}^{n} x_i \log x_i \leq k \log \frac{n}{k}. \tag{4.29}$$

Thus, for each sequence $(i_1, \ldots, i_N) \in \{1,2,3\}^N$, we obtain

$$\begin{aligned}
&- \sum_{x \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}} P[X = x] \cdot \log P[X = x] \leq \\
&= P[X \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}] \cdot \log \frac{|P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}|}{P[X \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}]} \\
&= \sum_{x \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}} \frac{P[X \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}]}{|P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}|} \cdot \log \frac{|P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}|}{P[X \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}]} \\
&= - \sum_{x \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}} P[X' = x] \cdot \log P[X' = x].
\end{aligned} \tag{4.30}$$

Using the inequality above, the desired result comes from

$$\begin{aligned}
H(X') &= - \sum_{i_1, \ldots, i_N} \sum_{x \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}} P[X' = x] \cdot \log P[X' = x] \\
&\geq - \sum_{i_1, \ldots, i_N} \sum_{x \in P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}} (P[X = x] \cdot \log P[X = x] \\
&= H(X).
\end{aligned} \tag{4.31}$$

□

This theorem shows that the estimation $\mathcal{D}^H$ gives the same probability to elements of $\mathcal{U}_\eta$ which the knowledge of the attacker does not distinguish. This is a desirable property, since it indicates that in some sense $\mathcal{D}^H$ does not give the attacker more information than what he can infer from his knowledge. Another reason why this theorem is important is that it provides us with a (slightly) simpler way to compute the distribution $\mathcal{D}_\eta^H$.

Corollary 4.1.8. *Consider the set of cryptographic properties $\mathbf{R}_\eta = \{r_1, \ldots, r_N\}$. Suppose that*

$$p = (p_k)_{k \in \{1,2,3\}^N}, \tag{4.32}$$

is a maximum of the function

$$\sum_{i_1=1}^{3} \ldots \sum_{i_N=1}^{3} |P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}| \cdot p_{(i_1,\ldots,i_N)} \log p_{(i_1,\ldots,i_N)} \tag{4.33}$$

restricted to

$$\begin{cases} \sum_{i_1=1}^{3} \ldots \sum_{i_N}^{3} |P_{r_1}^{i_1} \cap \ldots \cap P_{r_N}^{i_N}| \cdot p_{(i_1,\ldots,i_N)} = 1 \\ p_{(i_1,\ldots,i_N)} \geq 0. \end{cases} \tag{4.34}$$

Then the distribution $\mathcal{D}$ defined by

$$X \leftarrow \mathcal{D} \Rightarrow P[X = x] = p_{(i_1,\ldots,i_N)}, \tag{4.35}$$

where each i_j is such that $x \in P_{r_j}^{i_j}$, is a distribution of maximum entropy for $\mathbf{R}_\eta$.

The problem of maximizing a function in a set bounded by linear conditions is well studied in linear programming, and there are efficient algorithms for solving it. Of course, since the set $\{(i_1, \ldots, i_N) \colon i_j \in \{1, 2, 3\}, j = 1, \ldots, N\}$ has size 3^N and N may grow as a function of η, this is still not an efficient way of estimating the distribution. Indeed, we will soon see that the problem of estimating a probability in the distribution of maximum entropy is NP-hard in terms of N (the number of cryptographic properties known by the attacker).

The next theorem gives a result similar to 4.1.7. Namely, it states that if the knowledge of the attacker does not reveal any relation between the random variables represented by two expressions, then in the distribution $\mathcal{D}_\eta^H$ those expressions represent independent random variables. This is a nice property, for it means that the estimation $\mathcal{D}_\eta^H$ does not create dependencies between random variables unless the attacker explicitly knows some non-trivial relation between those variables.

We will use the following definition:

Definition 4.1.9. *Let r be the cryptographic property*

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] = \rho. \tag{4.36}$$

The set $\mathbf{E}(r)$ of expressions involved in r is

$$\{E^1, \ldots, E^n, E^{*,1}, \ldots, E^{*,n^*}\}. \tag{4.37}$$

If $\mathbf{R}$ is a set of cryptographic properties, we generalize this definition in the natural way:

$$\mathbf{E}(\mathbf{R}) = \bigcup_{r \in \mathbf{R}} \mathbf{E}(r). \tag{4.38}$$

Theorem 4.1.10. *Let $\mathbf{R}_\eta = \mathbf{R}_\eta^1 \cup \mathbf{R}_\eta^2$, such that $\mathbf{E}(\mathbf{R}_\eta^1) \cap \mathbf{E}(\mathbf{R}_\eta^2) = \emptyset$.*

Suppose that $E \in \mathbf{E}(\mathbf{R}_\eta^1), F \in \mathbf{E}(\mathbf{R}_\eta^2)$.

Then, if $\mathcal{D}_\eta^H$ is a maximum entropy distribution for $\mathbf{R}_\eta$ and $(E, F) \leftarrow \mathcal{D}_\eta^H$,

$$P[E = b, F = b'] = P[E = b] \cdot P[F = b'] \tag{4.39}$$

for any $b, b' \in \mathbf{B}_\eta$. In other words,

$$(E, F) \leftarrow \mathcal{D}_\eta^H \Rightarrow E \perp F. \tag{4.40}$$

Proof. We will use the same technique used in the proof of Theorem 4.1.7.

First, let $\mathbf{Exp}_\eta = \mathbf{E}_1 \cup \mathbf{E}_2$, so that

- $\mathbf{E}_1 \cap \mathbf{E}_2 = \emptyset$;
- $\mathbf{E}(\mathbf{R}^1_\eta) \subseteq \mathbf{E}_1$, $\mathbf{E}(\mathbf{R}^2_\eta) \subseteq \mathbf{E}_2$;
- $\mathbf{E}_1 = \{E^1_1, \ldots, E^{n_1}_1\}$, $\mathbf{E}_2 = \{E^1_2, \ldots, E^{n_2}_2\}$.

We define a distribution $\mathcal{D}'$ by

$$\begin{aligned} &P[(E^1_1)' = b_1, \ldots, (E^{n_1}_1)' = b_{n_1}, (E^1_2) = b^*_1, \ldots, (E^{n_2}_2)' = b^*_{n_2}] = \\ &P[E^1_1 = b_1, \ldots, E^{n_1}_1 = b_{n_1}] \cdot P[E^1_2 = b^*_1, \ldots, E^{n_2}_2 = b^*_{n_2}], \end{aligned} \tag{4.41}$$

where $(E^1_1, \ldots, E^{n_1}_1, E^1_2, \ldots, E^{n_2}_2) \leftarrow \mathcal{D}$, $((E^1_1)', \ldots, (E^{n_1}_1)', (E^1_2)', \ldots, (E^{n_2}_2)') \leftarrow \mathcal{D}'$.

From this definition it is easy to see that if $\{E^1, \ldots, E^n\} \subseteq \mathbf{E}_1$ (or $\mathbf{E}_2$) then

$$\begin{aligned} &P[(E^1)', \ldots, (E^n)' = b^*_{n^*}] = \\ &P[E^1, \ldots, E^n = b^*_{n^*}] \end{aligned} \tag{4.42}$$

(though the computations are tedious). Thus, it is immediate to see that

$$\begin{aligned} &P[(E^1)' = b_1, \ldots, (E^{n_1})' = b_{n_1}, (E^1_2)' = b^*_1, \ldots, (E^{n_2}_2)' = b^*_{n_2}] = \\ &P[E^1_1 = b_1, \ldots, E^{n_1}_1 = b_{n_1}] \cdot P[E^1_2 = b^*_1, \ldots, E^{n_2}_2 = b^*_{n_2}] = \\ &P[(E^1_1)' = b_1, \ldots, (E^{n_1}_1)' = b_{n_1}] \cdot P[(E^1_2)' = b^*_1, \ldots, (E^{n_2}_2)' = b^*_{n_2}], \end{aligned} \tag{4.43}$$

which proves that $\mathcal{D}'$ verifies (4.39).

Now let $r \in \mathbf{R}^1_\eta$ be the cryptographic property

$$P[E^1_\eta = b_1, \ldots, E^n_\eta = b_n \mid E^{*,1}_\eta = b^*_1, \ldots E^{*,n^*}_\eta = b^*_{n^*}] = \rho. \tag{4.44}$$

To see that $\mathcal{D}'$ verifies r, note that

$$\begin{aligned} &P[(E^1)' = b_1, \ldots, (E^n)' = b^n \mid (E^{*,n^*})', \ldots (E^{*,n^*})' = b^*_{n^*}] = \\ &\frac{P[(E^1)' = b_1, \ldots, (E^n)' = b^n, (E^{*,n^*})', \ldots (E^{*,n^*})' = b^*_{n^*}]}{P[(E^{*,n^*})', \ldots (E^{*,n^*})' = b^*_{n^*}]}. \end{aligned} \tag{4.45}$$

Since $\mathbf{E}(r) \subseteq \mathbf{E}(\mathbf{R}^1_\eta)$ and $\mathbf{E}(\mathbf{R}^1_\eta) \cap \mathbf{E}(\mathbf{R}^2_\eta) = \emptyset$, it is clear from 4.42 that

$$\begin{aligned} &P[X'_{E^{*,n^*}}, \ldots X'_{E^{*,n^*}} = b^*_{n^*}] = \\ &P[X_{E^{*,n^*}}, \ldots X_{E^{*,n^*}} = b^*_{n^*}]. \end{aligned} \tag{4.46}$$

A similar reasoning shows that

$$\begin{aligned} &P[(E^1)' = b_1, \ldots, (E^n)' = b^n, (E^{*,n^*})', \ldots (E^{*,n^*})' = b^*_{n^*}] = \\ &P[E^1 = b_1, \ldots, E^n = b^n, E^{*,n^*}, \ldots E^{*,n^*} = b^*_{n^*}]. \end{aligned} \tag{4.47}$$

Thus, from (4.45), we obtain

$$\begin{aligned} P[(E^1)' = b_1, \ldots, (E^n)' = b^n \mid (E^{*,n^*})', \ldots (E^{*,n^*})' = b^*_{n^*}] = \\ P[E^1 = b_1, \ldots, E^n = b^n \mid E^{*,n^*}, \ldots E^{*,n^*} = b^*_{n^*}]. \end{aligned} \tag{4.48}$$

We conclude that $\mathcal{D}'$ verifies all properties in $\mathbf{R}^1_\eta$. Checking that $\mathcal{D}'$ also verifies properties in $\mathbf{R}^2_\eta$ may be done analogously.

We now need to proof that $\mathcal{D}'$ has greater entropy than $\mathcal{D}$. This is a consequence of Theorem 4.1.4. To see this, we note that $X = (X_1, X_2)$, $X_1 = (E^1_1, \ldots, E^{n_1}_1)$, $X_2 = (E^1_2, \ldots, E^{n_2}_2)$, and adopt an analogous notation for X'.

From 4.42 we see that the distribution of X_1 (resp. X_2) is the same as the distribution of X'_1 (resp. X'_2). Thus, using the properties of Shannon entropy, we obtain

$$\begin{aligned} H(X) &= H(X_1, X_2) \\ &\leq H(X_2) + H(X_1) \\ &= H(X'_2) + H(X'_1) \\ &= H(X'_2) + H(X'_1 \mid X'_2) \\ &= H(X'_2, X'_1) \\ &= H(X') \end{aligned} \tag{4.49}$$

(where we use the fact that $X'_1 \perp X'_2$). □

These two theorems present nice properties of our distribution of maximum entropy. They show that our estimated distribution attributes the same probability to events which the attacker's knowledge does not distinguish and that it does not create dependencies between random variables other than those expressed by that knowledge. However, estimating this distribution is not simple. The next theorem shows that computing even one probability of this distribution is a NP-hard problem (in terms of the number of rules in $\mathbf{R}$). First we need to introduce the problem **3SAT**, a satisfiability problem for propositional logic.

Definition 4.1.11. *Fix a set $P = \{p_i : i \in \mathbb{N}\}$ of propositional symbols. A* literal *is a propositional formula consisting of either a propositional symbol (p_i) or the negation of a propositional symbol ($\neg p_i$). A* clause *is a formula of the form $l_1 \vee \ldots \vee l_n$, where each l_i is a literal (*i.e.*, a clause is a disjunction of literals).*

The problem **3SAT** *consists of, given a formula* ϕ *of the form*

$$\phi = (l_{11} \vee l_{12} \vee l_{13}) \wedge \ldots \wedge (l_{n1} \vee l_{n2} \vee l_{n3}), \tag{4.50}$$

where each l_{ij} *is a literal (*i.e.*,* ϕ *is a conjunction of clauses with three literals), determining whether or not there is an assignment of truth values to the symbols in* P *for which the formula* ϕ *is true.*

It is well-known that the problem **3SAT** is NP-complete [18]. The following theorem shows that **3SAT** can be reduced to the problem **Prob** of computing a value

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1} = b_1^*, \ldots, E^{*,n^*} = b_{n^*}^*] \tag{4.51}$$

where $(E^1, \ldots, E^n, E^{*,1}, \ldots, E^{*,n^*}) \leftarrow \mathcal{D}_\eta^H$ and $\mathcal{D}_\eta^H$ is a distribution of maximum entropy for a set of rules $\mathbf{R}$.

Theorem 4.1.12. *Let* **Prob** *be the following problem: given finite sets of expressions* **Exp***, of bitstrings* **B** *and of rules* **R***, estimate*

$$P[E^1 = b_1, \ldots, E^n = b_n \mid E^{*,1}, \ldots, E^{*,n^*}] \tag{4.52}$$

where $(E^1, \ldots, E^n, E^{*,1}, \ldots, E^{*,n^*}) \leftarrow \mathcal{D}_\eta^H$ *and* $\mathcal{D}_\eta^H$ *is a distribution of maximum entropy for the sets* **R***,* **Exp***,* **B***.*

The problem **3SAT** *reduces to* **Prob***. In particular,* **Prob** *is NP-hard (in* $|\mathbf{R}|$*).*

Proof. Let

$$\varphi = (l_{11} \vee l_{12} \vee l_{13}) \wedge \ldots \wedge (l_{n1} \vee l_{n2} \vee l_{n3}) \tag{4.53}$$

be an instance of the problem **3SAT** (*i.e.*, a propositional formula consisting of a conjunction of disjunctions of three literals). Denote by $S(\varphi) = \{s_1, \ldots, s_k\}$ the set of propositional symbols present in φ.

We will set

$$\begin{aligned} \mathbf{B} &= \{0, 1\} \\ \mathbf{E} &= \{E^{s_1}, \ldots, E^{s_k}, E^{c_1}, \ldots, E^{c_n}, E^{\varphi}\}. \end{aligned} \tag{4.54}$$

Intuitively:

- E^{s_i} represents the truth value of s_i: 1 if s_i is true, 0 if it is false;
- E^{c_i} represents the truth value of the i-th clause $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$;
- E^{φ} represents the truth value of the formula $\varphi = c_1 \wedge \ldots \wedge c_n$.

To describe the set $\mathbf{R}$ we will use the following notation:

- $s_{ij} \in S(\phi)$ is such that either $l_{ij} = s_{ij}$ or $l_{ij} = \neg s_{ij}$

- $b_{ij} = 1$ if $l_{ij} = s_{ij}$ and 0 if $l_{ij} = \neg s_{ij}$.

The set of rules $\mathbf{R}$ describes how the values of each E^{c_i} may be computed from the values of expressions $E^{s_1}, \ldots, E^{s_k}$ representing the truth values of propositional symbols $s_1, \ldots, s_k$. Thus, for each $i \in \{1, \ldots, n\}$, $\mathbf{R}$ contains the properties

$$\begin{cases} P[E^{c_i} = 1 \mid E^{s_{ij}} = b_{ij}] \text{ for } j = 1, 2, 3; \\ P[E^{c_i} = 0 \mid E^{s_{i1}} = 1 - b_{i1}, \ldots, E^{s_{i3}} = 1 - b_{i3}] \end{cases} \tag{4.55}$$

Furthermore, $\mathbf{R}$ also contains rules which describe how to compute E^{φ} from $E^{c_i}, i = 1, \ldots, n$:

$$\begin{cases} p[E^{\varphi} = 0 \mid E^{c_i} = 0] \text{ for } i = 1, \ldots, n; \\ p[E^{\varphi} = 1 \mid E^{c_1} = 1, \ldots, E^{c_n} = 1] \end{cases} \tag{4.56}$$

We will show that

$$\phi \text{ is satisfiable } \iff p^{est}[E^{\phi} = 1] > 0. \tag{4.57}$$

It is clear that the construction of the sets $\mathbf{B}, \mathbf{E}, \mathbf{R}$ may be done in time polynomial in the number n of clauses of the original formula: observe that $|\mathbf{B}| = 2$, $|\mathbf{E}| = k + n + 1 \leq 4n + 1$, $|\mathbf{R}| = 5n + 1$. Thus, the original instance of **3SAT** may be polynomially converted in this instance of **Prob**, and obtaining (4.57) concludes the demonstration.

In order to do this, we first note that each

$$x = (e^{s_1}, \ldots, e^{s_k}, e^{c_1}, \ldots, e^{c_n}, e^{\varphi}) \in \mathcal{U} \tag{4.58}$$

determines an assignment which attributes "true" to s_i if $e^{s_i} = 1$ and "false" otherwise.

Let $\mathcal{D}$ be a probability distribution in $\mathcal{U}$ which verifies the properties in $\mathbf{R}$. Suppose that $x = (e^{s_1}, \ldots, e^{s_k}, e^{c_1}, \ldots, e^{c_n}, e^{\varphi})$. Fix $i \in \{1, \ldots, n\}$. (4.55) tells us that

$$\begin{aligned} &e^{s_{ij}} = b_{ij} \text{ for some } j \in \{1, 2, 3\} \Rightarrow e^{c_i} = 1, \\ &e^{s_{ij}} = 1 - b_{ij} \text{ for all } j \in \{1, 2, 3\} \Rightarrow e^{c_i} = 0 \end{aligned} \tag{4.59}$$

provided that $x \in \mathcal{U}$ is sampled from $\mathcal{D}$ with non-zero probability: *i.e.*, $P[X = x] > 0$, $X \leftarrow \mathcal{D}$. This means that $e^{c_i} = 1$ if c_i is true for the assignment determined by x, and $e^{c_i} = 0$ otherwise. Similar considerations (using (4.56)) yield a similar conclusion for e^{φ}: $e^{\varphi} = 1$ if φ is true for the assignment determined by x and 0 otherwise.

Let $X_S = (E^{s_1}, \ldots, E^{s_k})$, $X_{C,\varphi} = (E^{c_1}, \ldots, E^{c_n}, E^{\varphi})$, so that $X = (X_S, X_{C,\varphi})$. From the reasoning above it is clear that $X_{C,\varphi}$ is determined by X_S. Thus, we have

$$\begin{aligned} H(X) &= H(X_S, X_{C,\varphi}) = \\ &= H(X_S) + H(X_{C,\varphi} \mid X_S) = \\ &= H(X_S). \end{aligned} \tag{4.60}$$

Thus, the distribution of maximum entropy $\mathcal{D}^H$ is such that X_S has uniform probability (when $X \leftarrow \mathcal{D}^H$).

(4.57) is now clear: if ϕ is satisfiable, then there is at least one assignment which satisfies it, and if x_S is the value of X_S which induces that assignment, then

$$P[X_{E^\phi} = 1] \geq P[X_S = x_s] = \frac{1}{2^k} > 0. \tag{4.61}$$

On the other hand, if f ϕ is not satisfiable, then for all x_S we have $P[X_{E^\phi} = 1, X_S = x_S] = 0$, and hence

$$P[X_{E^\phi} = 1] = \sum_{x_S} P[X_{E^\phi} = 1, X_S = x_S] = 0. \tag{4.62}$$

□

This theorem states that estimating this probability is NP-hard in the number of cryptographic properties known by the attacker. Note that this does not imply that the problem is NP-hard in the security parameter, for it is possible that the $\mathbf{R}_\eta$ does not grow when we increase the security parameter.

4.2 Examples

In this next section we present some examples of how to estimate probability distributions of random variables represented by expressions using the knowledge of the attacker, and further illustrate how we may use cryptographic properties to describe this knowledge.

The next example is a very simple but enlightening one. It consists of a protocol in which an attacker gradually reveals information about a secret message. This protocol is not secure at all; however, it does show how we may represent partial knowledge about a secret message in our model.

Example 4.2.1 (Duck-Duck-Goose Protocol)**.** *The Duck-Duck-Goose [17] is a very simple and utterly insecure protocol mentioned by Halpern and Pucella to illustrate the limitations of modelling attackers without cryptographic knowledge. It may be described as follows: an honest agent produces a (random) bitstring which is (supposedly) meant to be a secret. Then, the same agent reveals the bits composing that message one by one.*

To model this protocol we will use a random generation algorithm R with which the agent generates the "secret" bitstring. For simplicity we will assume that, for security parameter η, the secret message has exactly η bits. For example, $R(\eta)$ may generate a sequence of η random bits.

We will also need an algorithm B which, given $b \in \mathbf{B}$ and $k \in \mathbb{N}$, returns the k-th bit of b. More precisely, for each bitstring $b = (b_1 \ldots, b_\eta) \in \mathbf{B}_\eta$ $(b_1, \ldots, b_\eta \in \{0, 1\})$ and each $k \leq \eta$, $B(\eta, b, k) = b_k$. For simplicity, we will abbreviate $B(\eta, E, k)$ by E_k.

Furthermore, we need another algorithm G to allow the attacker to build a bitstring from the information he has obtained. This is a probabilistic algorithm; we will use algorithm R to model the randomness involved in this computation too. To compute

$$G(\eta, (b_1, k_1), \ldots, (b_n, k_n), r), r \leftarrow R,$$

the attacker "fills" the slots $k_1, \ldots, k_n$ with bits $b_1, \ldots, b_n$ and then fills the remaining $(\eta - n)$ slots with the first $(\eta - n)$ bits of r. Intuitively, the attacker feeds G with his knowledge of a message and G returns one possible message consistent with the information received as input.

The main cryptographic property which the attacker may compute is

$$p_\eta[G((E_{k_1}, k_1), \ldots, (E_{k_n}, k_n), R^j) = E] = 2^{n-\eta}, \tag{4.63}$$

where $E \in \mathbf{Exp}_\eta$, $k_i < \eta$ for all i and $k_i \neq k_j$ for all $i \neq j$.

The rules used to describe this protocol are also simple: for security parameter η, and each $k \leq \eta$, we have

$$\frac{(DDG, s, 1, A, R_1^i), \ldots, (DDG, s, k-1, A, R_{k-1}^i) \in \mathcal{H}}{\mathcal{H} := \mathcal{H} \cup \{(DDG, s, k, A, R_k^i)\}}. \tag{4.64}$$

A successful attack is when an instance of the protocol has started its execution and the attacker finds the secret message generated for that instance of the protocol. Thus, an attack is characterized by the condition

$$\begin{gathered}(DDG, s, 1, A, R_1^i) \in \mathcal{H} \\ (R_i = r^i) \in \mathcal{K}\end{gathered} \tag{4.65}$$

Using the same conventions already used in 3.2.1, 3.2.2 for simpler representation of attacks, an attack to this protocol (for security parameter η) may be written as follows.

1. $(DDG, s, 1, A, R_1^1)$
2. $R_1^1 = b_1$

   $\ldots$

$2\eta - 1$. $(DDG, s, \eta, A, R_\eta^1)$

2η. $R_\eta^1 = b_\eta$

$2\eta + 1$. $R^2 = r$

$2\eta + 2$. $G((R_1^1, 1), \ldots, (R_\eta^1, \eta), R^2) = g$

$2\eta + 3$. $R^1 = g$.

The last step is justified by (3.13)*, since, according to* (4.63)*,*

$$p_\eta[G((R_1^1, 1), \ldots, (R_1^1, \eta)) = R^1] = 1. \tag{4.66}$$

It is interesting to notice that after the $2k$*-th first steps of the protocol the attacker may try to guess the message, with probability of success* $2^{\eta-k}$*. Estimating probabilities like these is one of the main objectives of this probabilistic approach.*

Another interesting aspect about this example is estimating the probability distribution using the knowledge of the attacker (so far we have given him only equations of the type (4.63)*). This equation alone does not describe a significant relation between each* b_i *and the bitstring* b*: it merely tells us that a bitstring computed by* F *using* b_i *has a certain probability of corresponding to* b*. This is little information about* b*. For instance, equation* (4.63) *does not allow him to compute the probability* $P[R^1 = b \mid B(R_k^1) = b_k]$*. What the attacker can do in this case is compute* $G((b_k, k), R)$ *and estimate the probability distribution using the result of this computation, thus obtaining*

$$P[R^1 = b \mid G((b_k, k), R) = g] = \begin{cases} 2^{1-\eta} & \text{if } b = g \\ \frac{1-2^{1-\eta}}{2^\eta - 1} & \text{otherwise} \end{cases} \tag{4.67}$$

To better represent the knowledge of the attacker we should then include a rule to allow him to estimate the probability of $R^1 = b$ *given his knowledge of individual bits:*

$$P[R^1 = r \mid R_1^1 = b_1, \ldots, R_n^1 = b_n] = \begin{cases} 2^{n-\eta} & \text{if } r_{k_i} = b_i \text{ for } i = 1, \ldots, n \\ 0 & \text{otherwise} \end{cases} \tag{4.68}$$

The next example presents a popular game called Mastermind.

Example 4.2.2 (Mastermind)**.** *In a typical game of Mastermind, one of the players creates a code* C *by using one of eight available colors in each of four slots. The goal of the other player is to guess this code. For this, he may query the first player by presenting him a code* C' *built in the same fashion. The answer of the first player to this query is a pair of numbers* (R, W)*:* R *is the number of slots in which* C' *matches* C*, and* W *is the number of slots in* C' *filled with colors which were also used in* C*, but not in the same slot.*

For the purposes of our example we will consider that for each security parameter η *the game uses* η *colors and* η *slots. Also, for practical reasons, we will use bitstrings instead of colors. Otherwise, it is the same as the original game.*

Analogously to the previous example, we will use a random generation algorithm C *and a deterministic algorithm* B*.* $C(\eta)$ *returns an element of* $\{1, \ldots, \eta\}^\eta$ *(*i.e.*, a valid code for security parameter* η*). Given a code* C *and a number* n*,* B *returns the bitstring used in the* n*-th slot of* C*.* B *is independent of the security parameter, although for security parameter* η *only codes of length* η *will be used. As before, we will abbreviate* $B(\eta, E, k)$ *by* E_k*.*

The algorithms used by the attacker to respond to queries are R and W. These algorithms are defined (and may be computed) as follows:

$$R(\eta, C, C') = |\{k\colon C_k = C'_k\}|, \tag{4.69}$$

$$W(\eta, C, C') = |\{k\colon C_k \neq C'_k, \exists_{k'\neq k} C'_{k'} = C_k\}|. \tag{4.70}$$

The protocol may be described simply by two rules:

$$\frac{\begin{array}{c}(MM, s, 1, A, C_1), (MM, s, 2, B, (R(C_1, C^i), W(C_1, C^i))),\\ \ldots,\\ (MM, s, 2n-1, A, C_n), (MM, s, 2n, B, (R(C_n, C^i), W(C_n, C^i))) \in \mathcal{H}\end{array}}{(MM, s, 2n+1, A, C_{n+1})} \tag{4.71}$$

and

$$\frac{\begin{array}{c}(MM, s, 1, A, C_1), (MM, s, 2, B, (R(C_1, C^i), W(C_1, C^i))),\\ \ldots,\\ (MM, s, 2n-1, A, C_n) \in \mathcal{H}\end{array}}{(MM, s, 2n+1, B, (R(C_{n+1}, C^i), W(C_{n+1}, C^i)))}. \tag{4.72}$$

Here, $C_1, \ldots, C_n$ are any expressions, while C^i represents the code generated by the attacker. As usual, we require the i in C^i to be fresh in the second step of the protocol. Also observe that we have allowed agent B to reveal the bitstrings corresponding to two different expressions in the same step. This simplifies the exposition and avoids the need for pairing algorithms. Another solution would be to include two steps instead of one, the first revealing $R(C_i, C)$ and the second revealing $W(C_i, C)$.

Mastermind is an interesting example because we may obtain many different and possibly relevant cryptographic properties. Combining the answers of the "honest" player (the player who builds the code) with such cryptographic properties allows the attacker (the player who tries to guess the code) to find the secret bitstring (the code) in few steps. It is interesting (but hard) to study the probability distribution of maximum entropy resulting from these cryptographic properties. In here we describe very simple properties (in particular, we could as well ignore the function W) to show that the code may be broken in a polynomial number of queries.

Our attacker constructs messages by specifying the number in each slot and using an algorithm G. Thus, $G(\eta, \bar{k_1}, \ldots, \bar{k_\eta})$ (where $k_i \in \{1, \ldots, \eta\}$ for each i) is a valid code for security parameter η. G verifies

$$p_\eta[B(G(k_1, \ldots, k_\eta), i) = k_i] = 1. \tag{4.73}$$

Another property, analogous to (4.63)*, will also be used. Let R_c be a random algorithm used by the attacker to generate a random number (representing a color) such that $R_c(\eta)$ has*

uniform probability distribution in $\{1, \ldots, \eta\}$. *Our property then becomes*

$$p_\eta[G(E_1, \ldots, E_n) = C^i] = \eta^{k-\eta}, \tag{4.74}$$

where $E_{i_1}, \ldots, E_{i_k}$ *are* $C^i_{i_1}, \ldots, C^i_{i_k}$ *and the remaining* $(\eta - k)$ E_m *are randomly and independently generated from* R_c.

The main property used in our attack simply states that if two codes C', C'' *differ only in a slot* k *and* $R(C, C') = R(C, C'') + 1$, *then* $C_k = C'_k$:

$$p_\eta[C_k = b \mid R(C, C') = n + 1, R(C, C'') = n, C'_k = b] = 1. \tag{4.75}$$

Similarly to the previous example, an attack to this "protocol" consists of finding the bitstring corresponding to the code generated for a session of the protocol. Formally,

$$\begin{gathered}(MM, s, 1, A, C_1), (MM, s, 2, B, (R(C_1, C^i), W(C_1, C^i))) \in \mathcal{H} \\ (C = r^i) \in \mathcal{K}\end{gathered} \tag{4.76}$$

We need to include the second step of the protocol because the code is only generated in the second step. Of course, we assume here that A *is the agent name of the attacker.*

Our attack may then be described by the following function. Since the actions of the attacker in this case may depend on the results of previous computations, it is not practical to describe this attack in certificate form (of course, each possible trace of execution could be represented as such).

1. *For* $i = 1, \ldots, \eta$: $R^i_c \leftarrow R_c$
2. $C_0 \equiv G(R^1_c, \ldots, R^\eta_c)$
3. $(MM, s, 1, A, C_0)$
4. $(MM, s, 2, B, (R(C_0, C^k), W(C_0, C^k)))$ *(*k *fresh)*
5. $r_0 \leftarrow R(C_0, C^k)$
6. *For* $i = 1, \ldots, \eta$:
   (a) *For* $j = 1, \ldots, \eta$:
      i. $C_{i,j} \equiv G(R^1_c, \ldots, R^{i-1}_c, j, R^{i+1}_c, \ldots, R^\eta_c)$
      ii. $(MM, s, 2((i-1)\eta + j + 1) - 1, A, C_{i,j})$
      iii. $(MM, s, 2((i-1)\eta + j + 1), B, (R(C_{i,j}, C^1), W(C_{i,j}, C^k)))$
      iv. $r_{i,j} \leftarrow R(C_{i,j}, C^k)$
      v. *If* $r_{i,j} = r_0 + 1$ *Then* $C^1_i \leftarrow j$ *Else* $C^1_i \leftarrow R^j_c$
7. $C = G(C^1_1, \ldots, C^1_\eta)$.

In particular, an attacker may attack the Mastermind "protocol" with a number of actions of the order of η^2.

4.3 An Asymptotic Upper Bound on the "Observed" Entropy of Insecure Expressions

In this section we consider a polynomial-time attacker whose goal is to obtain the bitstring represented by a certain expression E. We present a result which may be useful in showing that a certain amount of cryptanalytical knowledge is not enough for one such attacker to have a non-negligible probability of success. The result concerns the (asymptotic) entropy of E given the attacker's guess.

Assume fixed a description of the public algorithms $\mathcal{F}, \mathcal{R}$ used by the agents in a given environment, and as before abbreviate $\mathbf{Exp} = \mathbf{Exp}(\mathcal{F}, \mathcal{R})$. In this section we will consider a polynomial-time attacker whose goal is to guess the bistring corresponding to a given expression. Let $E \in \mathbf{Exp}$ be that expression, and, for each η, let $\mathcal{A}(E_\eta)$ be the set of possible values of the corresponding random variable E_η, as before. We will assume two things about $\mathcal{A}(E_\eta)$:

- there are efficient algorithms for deciding whether or not a given bitstring is an element of $\mathcal{A}(E_\eta)$: *i.e.*, there is a (deterministic) polynomial-time algorithm e such that, for $\eta \in \mathbb{N}$, $b \in \mathbf{B}$,
$$e(\eta, b) = \begin{cases} 1 & \text{if } b \in \mathcal{A}(E_\eta) \\ 0 & \text{otherwise.} \end{cases} \tag{4.77}$$
- $|\mathcal{A}(E_\eta)|$ grows exponentially as a function of η: more precisely, there are $N \in \mathbb{N}, k > 1$ satisfying
$$\eta > N \Rightarrow |\mathcal{A}(E_\eta)| > k^\eta. \tag{4.78}$$

Intuitively, this last condition demands that $\mathcal{A}(E_\eta)$ is so large that guessing the bistring corresponding to E_η is "hard".

Assuming that there is an attacker A with a non-negligible probability of correctly guessing the bitstring corresponding to E_η, we present an attacker A' whose probability of success is the same as that of attacker A. A' may have additional cryptanalytical capabilities and may use a different set of deterministic and random generation algorithms: denoting by $\mathcal{F}^A$ (resp. $\mathcal{R}^A$) the set of deterministic (resp. random generation) algorithms available to the attacker A and by $\mathcal{F}^{A'}$ (resp. $\mathcal{R}^{A'}$) the set of deterministic (resp. random generation) algorithms available to A', we thus have $\mathcal{F}^A \subseteq \mathcal{F}^{A'}$, $\mathcal{R}^A \subseteq \mathcal{R}^{A'}$. We will write $\mathbf{Exp}^A \equiv \mathbf{Exp}(\mathcal{F}^A, \mathcal{R}^A)$, $\mathbf{Exp}^{A'} \equiv \mathbf{Exp}(\mathcal{F}^{A'}, \mathcal{R}^{A'})$. The set $\mathbf{B}'_\eta$ of possible values of a random variable E_η for $E \in \mathbf{Exp}_\eta^{A'}$ may also be different than the set $\mathbf{B}_\eta$ of possible values of a random variable $E_\eta \in \mathbf{Exp}_\eta^A$.

As we have also done in the previous sections, let

$$\mathcal{U}_\eta = \prod_{E \in \mathbf{Exp}_\eta^A} \mathbf{B}_\eta \tag{4.79}$$

and

$$\mathcal{U}'_\eta = \prod_{E \in \mathbf{Exp}_\eta^{A'}} \mathbf{B}'_\eta. \tag{4.80}$$

Also as we have done before, we denote by $\mathcal{D}_\eta$ the "real" probability distribution in $\mathcal{U}_\eta$: *i.e.*, the probability distribution $\mathcal{D}_\eta$ such that, if $X \leftarrow \mathcal{D}_\eta$ and $x \in \mathcal{U}_\eta$, then

$$\begin{aligned} P[X = x] = P[E_\eta = x_E \text{ for all } E \in \mathbf{Exp}_\eta^A] = \\ = P[E_\eta^1 = x_{E^1}, \ldots, E_\eta^n = x_{E^n}] \end{aligned} \tag{4.81}$$

where $\{E^1, \ldots, E^n\} = \mathbf{Exp}_\eta^A$. Similarly, we define $\mathcal{D}'_\eta$ as the "real" probability distribution in $\mathcal{U}'_\eta$, so that, if $X \leftarrow \mathcal{D}'_\eta$ and $x \in \mathcal{U}'_\eta$, we have

$$\begin{aligned} P[X = x] = P[E_\eta = x_E \text{ for all } E \in \mathbf{Exp}_\eta^{A'}] = \\ = P[E_\eta^1 = x_{E^1}, \ldots, E_\eta^N = x_{E^N}] \end{aligned} \tag{4.82}$$

where $\{E^1, \ldots, E^N\} = \mathbf{Exp}_\eta^{A'}$. It is easy to see that if $E \in \mathbf{Exp}_\eta^A$ then $E \in \mathbf{Exp}_\eta^{A'}$, and the corresponding random variables have the same probability distribution. Thus, if $X \leftarrow \mathcal{D}_\eta$ and $X' \leftarrow \mathcal{D}'_\eta$, we have

$$b \in \mathbf{B}_\eta \Rightarrow P[X_E = b] = P[X'_E = b], \tag{4.83}$$

and

$$b \in \mathbf{B}'_\eta \setminus \mathbf{B}_\eta \Rightarrow P[X'_E = b] = 0. \tag{4.84}$$

A' has two important properties. First, the set $\mathcal{K}$ which represents A''s knowledge by the time it takes its guess (*i.e.*, at the end of the attack) may be computed by a polynomial-time algorithm K which depends on the security parameter and on the bitstrings sampled from random generation algorithms during the attack (either by honest agents or by the attacker). The second property is that A''s guess may be computed from that set of equalities $\mathcal{K}$ by a (deterministic) polynomial-time algorithm g.

Now the set $\mathcal{R}^{A'}$ is finite and contains the random generation algorithms available to both the attacker and the honest agents; furthermore, each random generation algorithm may only be sampled at most a polynomial number of times, say η^k during the attack. Thus, if we write $\mathcal{R}^{A'} = \{R_1, \ldots, R_n\}$, then for a given security parameter η the set $\Omega_\eta = R_1^1, \ldots, R_1^{\eta^k}, \ldots, R_n^1, \ldots, R_n^{\eta^k}$ contains all random expressions in $\mathcal{U}'_\eta$ whose value may be (randomly) sampled during the execution of the attack. Using Definition 2.2.3, we may write

$$\bigcup_{E \in \mathbf{Exp}_\eta^{A'}} random(E) \subseteq \mathbf{Exp}_\eta^{A'}, \tag{4.85}$$

and

$$\bigcup_{E \in \mathbf{Exp}_\eta^{A'}} random(E) \subseteq \Omega_\eta. \tag{4.86}$$

We thus equip A' with a polynomial-time deterministic algorithm $K \in \mathcal{F}^{A'}$ such that

$$K(\eta, \Omega_\eta \equiv K(\eta, R_1^1, \ldots, R_1^{\eta^k}, \ldots, R_n^1, \ldots, R_n^{\eta^k}) \tag{4.87}$$

(where we assume, without loss of generality, that $\Omega_\eta \subseteq \mathbf{Exp}_\eta^{A'}$) is a bitstring representing the knowledge of the attacker at the end of the attack given the randomly sampled data during the execution of the protocol.

Since A''s guess may be deterministically computed in polynomial-time from the set $\mathcal{K}$, we also consider an algorithm $g \in \mathcal{F}^{A'}$ such that, if the values for each random expression sampled during the execution of the protocol verify

$$R_1^1 = r_1^1, \ldots, R_1^{\eta^k} = r_1^{\eta^k}, R_n^1 = r_n^1, \ldots, R_n^{\eta^k} = r_n^{\eta^k}, \tag{4.88}$$

then

$$g(\eta, K(\eta, r_1^1, \ldots, r_1^{\eta^k}, r_n^1, \ldots, r_n^{\eta^k})) \tag{4.89}$$

is the guess of the attacker.

Our result states that

$$H_{max}(E_\eta) - H(E \mid g(K(\Omega_\eta))), \tag{4.90}$$

where $E, g(K(\Omega_\eta)) \leftarrow \mathcal{D}_\eta^H$ and $\mathcal{D}_\eta^H$ is the distribution of maximum entropy on $\mathcal{U}_\eta'$ which verifies the cryptanalytical properties known by A', is non-negligible as a function of η.

In this expression,

$$H_{max}(E_\eta) = \log|\mathcal{A}(E_\eta)| \tag{4.91}$$

is the maximum possible entropy for a random variable with the same set of possible values as E_η; it is obtained when E_η has uniform probability distribution in $\mathcal{A}(E_\eta)$. The uniform distribution may also be interpreted as the distribution of maximum entropy of E_η for an attacker without any crypatanalytical knowledge of E_η which gives a "dummy" random guess; thus, we may intuitively interpret this theorem as stating that the knowledge of the attacker at the end of the attack (including both cryptanalytical knowledge and partial knowledge about the messages exchanged in the network) non-negligibly reduces the entropy of E_η.

We now state this theorem and present a sketch of its proof.

Theorem 4.3.1. *Let $E \in$ **Exp** be such that there is a polynomial-time algorithm e and $N \in \mathbb{N}, k > 1$ satisfying* (4.77), (4.78)*. Suppose that there is a polynomial-time attacker A whose probability of guessing the bitstring represented by E_η is a non-negligible computable function of η.*

There is another attacker A' and polynomial-time algorithms $g, K \in \mathcal{F}^{A'}$ such that:

- *for each η, if* (4.88) *holds then* (4.89) *is A''s guess, and*
- $$H_{max}(E_\eta) - H(E \mid g(, K(\Omega_\eta))), \tag{4.92}$$

  where $E, g(K(\Omega_\eta)) \leftarrow \mathcal{D}_\eta^H$ and $\mathcal{D}_\eta^H$ is the distribution of maximum entropy on $\mathcal{U}'_\eta$ which verifies the cryptanalytical properties known by A', is non-negligible as a function of η.

Proof. The first step of the proof is to see that we may specify an attacker A' with the two properties described above.

For this, suppose that, whenever the attacker A would make a probabilistic decision (whether for choosing his next action or for some internal computation), A' runs some random generation algorithm $R \in \mathcal{R}^{A'} \setminus \mathcal{R}^A$ and stores the result of executing $R(\eta)$ in $\mathcal{K}$. The algorithm R is not used for any other purpose, and we assume that the i-th probabilistic decision of the attacker is represented by the bitstring corresponding to the expression R^i, so that by looking at the final set $\mathcal{K}$ one knows which expression corresponds to each decision. Clearly, we may assume that the probability of each decision is the same for both attackers, A and A', and so their probability of success is also the same. Note that A' essentially performs the same actions as A with the same probability; the main difference is that A' records each decision in the set $\mathcal{K}$, as the output of a random generation algorithm.

The algorithm K works as follows. It starts with empty sets $\mathcal{H}', \mathcal{K}'$, and mimics A's attack strategy. Whenever A' would make a probabilistic choice, K makes the choice determined by the value of the corresponding expression R^i. Each action of the attacker consists of either adding an element to the set $\mathcal{H}$ (corresponding to the publication of a message) or adding an element to the set $\mathcal{K}$ (in which case the attacker learns the value of some expression $E \in \mathbf{Exp}_\eta^{A'}$).

In the first case, the element added to $\mathcal{H}$ is determined by the attacker (including the structure of the message published, the attacker who publishes it, the protocol session to which it corresponds, and so on) based on its internal computations. Thus, algorithm K simply adds that same element to the set $\mathcal{H}'$, taking into account the probabilistic decisions made by A and stored as the values of the expressions R^i, $i = 1, \ldots$.

Regarding the second case, our definitions imply that the value of each expression in $\mathbf{Exp}_\eta^{A'}$ may be computed in polynomial-time from the value of its random subexpressions, since each algorithm in $\mathcal{F}^{A'}$ is a polynomial-time deterministic algorithm. Thus, if A would add an equality $(E^* = b)$ to $\mathcal{K}$, algorithm K simply adds that equality to the set $\mathcal{K}'$. This involves, at most, computing the value of the expression E^* from the values of the expressions in $random(E^*)$, which are given as input to K. The algorithm ends when A would take its guess, and K returns the set $\mathcal{K}'$ in its current state.

The algorithm g follows a similar procedure. We should just note that, given the set $\mathcal{K}$, the algorithm g may simulate the trace of attack which lead to that set $\mathcal{K}$, since it includes the values of random expressions which determine A's probabilistic decisions as well as all the knowledge which A would have at each step of the attack. Thus, g may compute A''s guess (which is the same as A's guess) from η and $K(\Omega_\eta)$.

A''s cryptanalytical knowledge contains the property

$$p_\eta[Equals(E, g(K(\Omega_\eta))) = 1] = s(\eta), \tag{4.93}$$

where $Equals \in \mathcal{F}^{A'}$ is an algorithm which receives two bitstrings and returns 1 if they are equal, 0 otherwise; *i.e.*, for each η and each $b_0, b_1 \in \mathbf{B}'_\eta$,

$$Equals(\eta, b_0, b_1) = \begin{cases} 1 & \text{if } b_0 = b_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.94}$$

This property simply states that the probability that the guess of the attacker is correct is given by $s(\eta)$, and thus is correct by hypothesis. Furthermore, we let the attacker know that the value of E_η is in $\mathcal{A}(E_{eta})$: *i.e.*, we let $e \in \mathcal{F}^{A'}$, and include in the cryptanalytical knowledge of the attacker the properties

$$p_\eta[E = b \mid e(E) = 0] = 0, \tag{4.95}$$

for all $E \in \mathbf{Exp}_\eta^{A'}$.

We also need some properties which the algorithm $Equals$ to prove our result:

$$\begin{aligned} p_\eta[E' = b \mid E = b, Equals(E, E') = 1] = 1 \\ p_\eta[E' = b \mid E = b, Equals(E, E') = c] = 0, \end{aligned} \tag{4.96}$$

where $E, E' \in \mathbf{Exp}_\eta^{A'}$, $b \in \mathbf{B}'_\eta$ and $c \in \mathbf{B}'_\eta \neq 1$.

Now, given $g(K(\Omega_\eta)) = b$ for some b, these properties imply that

$$\begin{aligned} P[E_\eta = b] = s(\eta), \\ P[E_\eta \notin \mathcal{A}(E_\eta)]. \end{aligned} \tag{4.97}$$

when $E_\eta \leftarrow \mathcal{D}_\eta^H$ is sampled from the distribution of maximum entropy given A''s cryptanalytical knowledge. Without any further knowledge of E, it follows that, in the distribution of maximum entropy,

$$P[E = b'] = \frac{1 - s(\eta)}{|\mathcal{A}(E_\eta)| - 1} \tag{4.98}$$

for $b' \neq b$, $b' \in \mathcal{A}(E_\eta)$.

Thus,

$$\begin{aligned} H(E \mid g(K(\Omega_\eta))) &= -s(\eta)\log(s(\eta)) - (1-s(\eta))\log\frac{1-s(\eta)}{|\mathcal{A}(E_\eta)|-1} \\ &< -\frac{1}{\eta^c}\log\frac{1}{\eta^c} - (1-\frac{1}{\eta^c})\log\frac{1-\frac{1}{\eta^c}}{|\mathcal{A}(E_\eta)|-1}, \end{aligned} \tag{4.99}$$

for sufficiently large η and some $c > 0$ such that $s(\eta) > \frac{1}{\eta^c}$. By definition of H_{max},

$$H_{max}(E_\eta) = \log(|\mathcal{A}(E_\eta)|). \tag{4.100}$$

Now

$$\begin{aligned} H_{max}(E_\eta) - H(E \mid g(K(\Omega_\eta))) \geq & \log(|\mathcal{A}(E_\eta)|) \\ & + \frac{1}{\eta^c}\log\frac{1}{\eta^c} \\ & + (1-\frac{1}{\eta^c})\log\frac{1-\frac{1}{\eta^c}}{|\mathcal{A}(E_\eta)|-1} \\ = & \log(|\mathcal{A}(E_\eta)|) \\ & + \frac{1}{\eta^c}\log\frac{1}{\eta^c} \\ & + (1-\frac{1}{\eta^c})\log(1-\frac{1}{\eta^c}) \\ & - (1-\frac{1}{\eta^c})\log(|\mathcal{A}(E_\eta)|-1) \\ = & [\log(|\mathcal{A}(E_\eta)|) - \log(|\mathcal{A}(E_\eta)|-1)] \\ & + \frac{1}{\eta^c}\log\frac{1}{\eta^c} \\ & + (1-\frac{1}{\eta^c})\log(1-\frac{1}{\eta^c}) \\ & + \frac{1}{\eta^c}\log(|\mathcal{A}(E_\eta)|-1). \end{aligned} \tag{4.101}$$

and:

- $$\eta^{c-1}[\log(|\mathcal{A}(E_\eta)|) - \log(|\mathcal{A}(E_\eta)|-1)] < \frac{\eta^{c-1}}{|\mathcal{A}(E_\eta)|-1} \tag{4.102}$$

  goes to 0 when η goes to infinity, since we assumed that $|\mathcal{A}(E_\eta)| > k^\eta$ for some $k > 1$;
- $\eta^{c-1}\frac{1}{\eta^c}\log\frac{1}{\eta^c}$ and $\eta^{c-1}(1-\frac{1}{\eta^c})\log(1-\frac{1}{\eta^c})$ also go to 0 when η goes to infinity, applying properties of logarithms;
- since $|\mathcal{A}(E_\eta)| > k^\eta$ for some $k > 1$, we have

$$\log(|\mathcal{A}(E_\eta)|) > \eta\log k, \tag{4.103}$$

for all sufficiently large η, and thus

$$\eta^{c-1}\frac{\log(|\mathcal{A}(E_\eta)|-1)}{\eta^c} > C, \tag{4.104}$$

where $C = \log k > 0$ is some positive constant.

From the previous we conclude that

$$H_{max}(E_\eta) - H(E \mid g(K(\Omega_\eta))) > \frac{C'}{\eta^{c-1}} \tag{4.105}$$

for all sufficiently large η, which concludes the demonstration. □

Intuitively, this theorem says that the knowledge of a successful attacker should decrease the entropy of the secret bitstring non-negligibly. The specific cryptographic property and the alternative attacker used in this proof are not very interesting by themselves, but they do allow us to show that this intuition is correct. Observe however that our proof shows that for any attacker A we may construct an attacker A' which is essentially the same as the attacker A and whose guess may be written as $g(K(\Omega))$.

This theorem may be useful in proving that a certain set of cryptographic properties do not compromise the security of a protocol. In fact, suppose that (4.92) is negligible for an algorithm g representing some attacker's guess. If that attacker has a non-negligible probability of success, there must be some other relevant, polynomial-time computable property which give non-negligible information about E_η from the attacker's knowledge. If given the knowledge of the attacker there is no guessing algorithm g such that (4.92) is non-negligible, then the set of cryptographic properties which we allow the attacker to use do not compromise the security of the protocol by themselves. Of course, when we consider specific cryptographic primitives, they may verify other properties which (together with the properties described in the setting) allow an attacker to be successful with non-negligible probability.

Note also that the converse of the theorem is not true. The following two examples illustrate this.

Example 4.3.2. *Suppose that, for each η, E_η has uniform probability distribution in $\mathcal{A}(E_\eta) = \{0,1\}^\eta$. Then, $H_{max}(E_\eta) = \eta$ for each η. If during the attack the attacker is able to learn the first bit of E, represented by $bit(E)$. The attacker obtains its guess by randomly generating a bitstring s in $\{0,1\}^{\eta-1}$ and guessing $(bit(E), s)$. Given a guess (bit, s), the probability distribution of E_{eta} in the distribution of maximum entropy is then*

$$P[E_\eta = b] = \begin{cases} \eta^{c-1} & \text{if } bit(b) = bit \text{ and } b \in \{0,1\}^\eta \\ 0 & \text{otherwise} \end{cases} \tag{4.106}$$

Thus,

$$\begin{aligned} H_{max}(E_\eta) &= \eta, \\ H(E \mid g(K(\Omega_\eta))) &= \eta - 1 \end{aligned} \tag{4.107}$$

(for algorithms g, K as described in the proof of the previous theorem), so that 4.92 is non-negligible. However, the attacker's probability of success is $2^{\eta-1}$, and decreases exponentially in η.

The next example is perhaps more disturbing: it deals with an attacker whose probability of success is 0 (in most reasonable situations), but such that $H(E \mid g(K(\Omega_\eta)))$ is 0 for the algorithms g, K as described above.

Example 4.3.3. *Suppose that an attacker A wants to find the value of some expression E, and suppose that during the attack A obtains $\{E\}_{K,R}$ (the encryption of E with some public key K and randomly generated data R) and the public key K.*

We consider that the attacker knows the encryption algorithm Enc, and his cryptanalytical knowledge includes the cryptographic properties

$$p_\eta[\{E\}_{K,R} = Enc(\eta, b, k, r) \mid E = b, K = k, R = r] = 1 \tag{4.108}$$

for all η, all expressions $E, K, R \in \mathbf{Exp}_\eta^A$ and all valid bitstrings b, k, r.

If the attacker's guess is $(\{E\}_{K,R}, K)$, then

$$H(E \mid (\{E\}_{K,R}, K)) = 0, \tag{4.109}$$

since the values of $\{E\}_{K,R}$ and K uniquely determine the value of E in the distribution of maximum entropy. However, the probability that $(\{E\}_{K,R}, K) = E$ may be as low as 0 for most encryption schemes.

Chapter 5

Conclusions and Further Work

In this work we presented a formal model for analysing the security of cryptographic protocols against attackers who may use cryptanalysis. This model allows the representation of cryptanalytical capabilities and partial information about secret messages. It is a very general and flexible model: one may specifiy cryptographic properties and use them to find an attack, but if one does not it behaves essentially as a formal model.

We showed how a standard security notion (indistinguishability under chosen-plaintext attacks) may be expressed in terms of the cryptographic properties considered, and concluded that its usual statement may be replaced by a simpler one. We also pointed out a problem with this definition.

Finally, we proposed a way of estimating the attacker's probability of success. To obtain this estimation we define the probability distribution of maximum entropy based on known cryptographic properties. We present a few properties of this distribution which make it a reasonable approach to model an attacker with cryptanalytical knowledge. This technique may also be used to study whether a weakness of the cryptographic primitives used compromises or not the security of a given protocol. Another advantage of this is that it allows, to some extent, a separate study of weaknesses of cryptographic primitives and the security of protocols. In fact, the model verifies the security of a protocol against an attacker who may use certain cryptographic weaknesses. Thus, one may simply write all properties of the cryptographic primitives deemed relevant and verify if the protocol is still secure when the attacker explores these properties.

Several problems and open questions still remain. First of all, for the model to be effective, one needs to give a complete description of the cryptographic properties of the functions used, which should be usually quite hard. Thus, for each cryptographic primitive, finding a suitable set of properties to describe its weaknesses is a non-trivial and interesting problem. Finding a criteria for deciding which properties are relevant, at least in some specific protocol, is also an interesting problem.

Examples of attack using properties of real cryptographic primitives are tipically hard to manage, as the complexity of the computations involved make even writing properties a non-trivial exercise. Implementing the model should help overcome this difficulty. The task of finding an attack has exponencial complexity on the length of the attack, as is tipically

the case for most tools used for this task. For example, linear and differential cryptanalysis explore properties like the ones considered in this work, and one that could be studied with an automated version of our model. Another possible application may be found in side channel attacks, in which the attacker explores the physical implementation of algorithms to obtain partial information about secret data (such as a key).

Other interesting and relevant problem is to find alternative methods for estimating probability distributions of bitstrings based on the knowledge of the attacker. Though the proposal presented here has many advantages and is relatively natural, other methods may be more efficient or reflect better the power and knowledge of the attacker, given his cryptanalytical capabilities.

Our results about IND-CPA security also compel us to determine adequate security notions which may be used in proofs of protocol security, particularly within our framework. Other alternatives may use, for example, the distribution of maximum entropy given the knowledge of the attacker.

Overall, our work presents an alternative formal model which allows partial information about secret messages and cryptanalytical knowledge to be used to find attacks. It also raises many interesting questions and problems which we believe may constitute an interesting topic of further research.

Bibliography

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Proceedings 31st Int. Coll. Automata, Languages and Programming (ICALP 2004)*, pages 46–58, 2004.

[2] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Proc. 4th ACM Conference on Computer and Communication Security*, pages 36–47, 1997.

[3] M. Abadi and P. Rogaway. Reconciling two views of cryptography. *Journal of Cryptology*, pages 103–127, 2002.

[4] P. Adão. Formal methods for the analysis of security protocols. 2006.

[5] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, L. Cuellar, P. Drielsma, P. Heám, O. Kouchnareko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for Automated Validation of Internet Security Protocols and Applications. *Lecture Notes in Computer Science*, 3576, 2005.

[6] M. Backles and B. Pfitzmann. A cryptographically sound security proof of the needham-schröeder-lowe public-key protocols. *IEEE Journal on Selected Areas in Communications*, 2004.

[7] C. Caleiro, D. Basin, and L. Viganò. Deconstructing Alice & Bob. *Workshop on Automated Reasoning for Security Protocol Analysis*, 2005.

[8] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A metanotation for protocol analysis. *12-th IEEE Computer Security Foundations Workshop*, 1999.

[9] C. Cremers. Scyther - Semantics and verification of security protocols. 2006.

[10] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, pages 198,208, 1983.

[11] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1999.

[12] Taher Elgamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, pages 10–18, 1985.

[13] J. Y. Halpern and Y. Moses. Reasoning about knowledge. 1995.

[14] J. Y. Halpern, Y. Moses, and M. Y. Vardi. Algorithmic knowledge. *Proceedings 5th Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 255–266, 1994.

[15] J. Y. Halpern and R. Pucella. Probabilistic algorithmic knowledge. *Logical Methods in Computer Science*, 2005.

[16] R. Hilpinen. *Carnap's New System of Inductive Logic*, volume 25. Springer Netherlands.

[17] J.Y.Halpern and R. Pucella. modeling adversaries in a logic for reasoning about security protocols. *Proceedings Workshop on Formal Aspects of Security*, 2002.

[18] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*.

[19] P. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polynomial-time framework for protocol analysis. *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS)*, pages 112–121, 1998.

[20] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[21] L.C. Paulson. Proving properties of security protocols by induction. *10th IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.

[22] L. Adelman R. Rivest, A. Shamir. A method for obtaining digital signatures and public-key cryptosystem. *Communications of the ACM*, 1978.

[23] D. Song. Athena: a new efficient automatic checker for security protocol analysis. *Proceedings of the Twelfth IEEE Computer Security Foundations Workshop*, pages 192–202, 1999.

[24] D. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.

[25] B. Warinschi. A computational analysis of the Needham-Schröeder-(Lowe) protocol. *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW)*, pages 248–262, 2003.