A Computational Analysis of the Needham-Schröeder-(Lowe) Protocol

BOGDAN WARINSCHI*
Department of Computer Science and Engineering,
University of California, San Diego
9500 Gilman Drive, CA 92093
bogdan@cs.ucsd.edu

Abstract

We provide the first computational analysis of the well known Needham-Schröeder(-Lowe) protocol. We show that Lowe's attack to the original protocol can naturally be cast to the computational framework. Then we prove that chosen-plaintext security for encryption schemes is not sufficient to ensure soundness of formal proofs with respect to the computational setting, by exhibiting an attack against the corrected version of the protocol implemented using an ElGamal encryption scheme. Our main result is a proof that, when implemented using an encryption scheme that satisfies indistinguishability under chosen-ciphertext attack, the Needham-Schröeder-Lowe protocol is indeed a secure mutual authentication protocol. The technicalities of our proof reveal new insights regarding the relation between formal and computational models for system security.

1 Introduction

CONTEXT. It is fair to say that today's cryptographic research community is divided in two rather non-overlapping sub-communities, each with a history spanning more than twenty years. The sizable gap between the two communities is largely due to the use of different approaches for modeling security of systems, each approach having its own advantages and disadvantages.

The formal methods approach, initiated in [7] also known as the Dolev-Yao model, is aimed at making proving security of protocols a manageable task. To this end, by abstracting away the details of the primitives used to build up complex systems, researchers in this community are able to automate this kind of proofs by employing methodologies and tools specific to the formal methods community, such as model checkers and theorem provers [16, 21, 15]. However, the high degree of abstraction that characterizes this approach raises a legitimate question: what are the precise security guarantees that a formal proof entails? Although recently a significant amount of effort [1, 18, 12] was aimed at answering this question, no satisfactory answer is known so far.

The computational approach (also known as provable-security) was initiated in [11]. Its vocabulary is a mixture of probability and complexity theory. Cryptographic operations are viewed as algorithms operating on actual bit-strings, and more complex protocols are defined by combining the Turing machines running these algorithms. The stronger and more comprehensive adversarial model of this approach has gained wide acceptance in the cryptographic community as providing the "right" security model. However, the detailed treatment of cryptographic primitives and the lack of automation (due to the relatively unstructured way in which security is formalized in this framework), make proving security of even a moderately complex protocol quite a formidable and error prone task.

THE PROBLEM. The usefulness of the formal methods approach was conclusively demonstrated in 1995, when Gavin Lowe discovered a subtle bug in a cryptographic protocol [13] invented by R. Needham and M. Schröeder nearly *twenty years* before [17]. In a subsequent paper [14], the same author suggests a way to fix the protocol, and shows that the modified version is indeed secure.

The simplicity of the two protocols (they involve only public key encryption and nonce generation), combined with Lowe's results made the protocols *the* test case for both new and well established formal approaches to verifying security.

Surprisingly, despite its relevance for the formal methods community, until now, a computational analysis of the security of the Needham-Schröeder (hence forth NS) protocol has not been provided. We point out two reasons why such

*Research supported in part by NSF Career Award CCR-0093029; part of the work was done while the author was visiting the IBM-Zurich Research Laboratory

an analysis is an important step towards narrowing the gap between the formal and computational approaches.

Firstly, a negative result showing that instantiating the formal encryption with a concrete encryption scheme satisfying the strongest accepted security notion does not yield a secure protocol (from a computational perspective) would cast serious doubts on the soundness of formal proofs (with respect to computational models). Secondly, if the protocol is indeed secure, given the multitude of formal methods proofs for the security of this protocol, going into the "dirty" details of a complete computational proof may shed some light on the connections between the two worlds.

OUR RESULTS. We start by showing that the original protocol of R. Needham and M. Schröeder is insecure for any instantiation of the encryption scheme. This result is not novel and easily follows from Lowe's result, but our technique shows that formal attackers can be naturally translated into computational attackers, regardless of the inherent difference between the models. Next, we focus on Lowe's version of the protocol and show that his fix may not be sufficient (for the computational security requirement) even if the encryption scheme used in an implementation satisfies a standard notion of security. More precisely, we give an implementation for the classical ElGamal encryption scheme [8] satisfying indistinguishability under chosen-plaintext attack [11] and show that, by exploiting some algebraic properties of our specific implementation, one can mount a damaging attack against the protocol.

Our main result is the first proof of security, in the computational setting, for Lowe's version of the Needham-Schröeder protocol. We show that if the encryption scheme used to implement the protocol satisfies a strong privacy condition, namely indistinguishability under chosen-ciphertext attack [19], then the resulting instantiation of the protocol is indeed secure. While the result may not surprise, the intricate technical details of the proof reveal interesting observations regarding connections between formal methods proofs and computational ones. For instance, the way mutual authentication is modeled in the formal approach known as the strand space model [9] and the computational model that we use, are essentially the same. Our findings are presented in Section 6.

The above results trigger a natural question. Do there exist security notions for encryption, weaker than IND-CCA, sufficient to ensure the security of the NSL protocol? Although in this paper we do not provide a definitive answer to this question, a possible answer is suggested by the technical details of our proof. More precisely, it seems that the recently introduced notion of *benign malleability*[20], or the equivalent formulation of [2], termed gCCA security, better captures the intuition behind the Dolev-Yao-type encryption. We explain our findings in Section 6.

The rest of the paper is organized as follows. Section 2 contains background related to the computational approach. In Section 3 we show how to adapt the model of [5] for multi-party protocol execution to the public key setting. The following section (Section 4) is dedicated to introducing and formally defining the concept of mutual authentication in the computational framework. Our results concerning the security of the Needham-Schröeder protocol are given in Section 5. Section 6 concludes with a brief discussion of our findings.

2 Preliminaries

In this section we briefly recall some terminology and concepts part of the usual vocabulary of the complexity-based approach to cryptography. More details can be found in [10, 4]. The discussion is rather standard and it can be safely skipped by readers familiar with this approach.

We denote by $\{0,1\}^*$ the set of all finite length strings. The length of such a string is denoted by $|x|$. If D is a samplable distribution we denote by $x \stackrel{R}{\leftarrow} D$ the process of sampling x according to distribution D.

From a computational perspective, all parties that are involved in a cryptographic system, being honest participants or adversaries, are identified with algorithms taking strings as inputs and having strings as output. The size of the the data involved in the computations as well as the ability of the adversary to "break" a cryptographic system are measured in terms of a *security parameter*. A typical security requirement states that for any efficient adversary, the likelihood that the adversary "breaks" the system is extremely small provided that the security parameter is large enough. Usually, by "efficient" it is meant polynomial-time computable (in the security parameter), and by "very small" it is meant smaller than some negligible function (in the security parameter). Recall that a function $f(\cdot)$ is said to be negligible if it is smaller than the inverse of any polynomial, i.e for any polynomial p, there exists $k_0 \in \mathsf{N}$ such that $f(k) \leq \frac{1}{p(k)}$ for all $k \geq k_0$. We will exemplify this kind of security requirement in the context of asymmetric encryption schemes.

ASYMMETRIC ENCRYPTION SCHEMES. An *asymmetric encryption scheme* $\mathcal{AE}$ is given by a triple of algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ where

- Kg is the randomized *key generation algorithm*; on input a security parameter k outputs a public key pk together with a corresponding secret key sk; we write $(pk, sk) \stackrel{R}{\leftarrow} \mathsf{Kg}(k)$.
- Enc is the randomized *encryption algorithm*; it takes as input a public key pk and a *plaintext* m, and outputs a *ciphertext* c; we write $c \stackrel{R}{\leftarrow} \mathsf{Enc}_{pk}(m)$ for the process of generating a ciphertext c associated to m.

- Dec is the deterministic *decryption algorithm*; it takes as input a secret key sk and a ciphertext c and outputs the underlying plaintext; we assume that if the decryption does not succeed (the ciphertext was not valid) the decryption algorithm returns some error symbol, say $\perp$.

Any encryption scheme should satisfy the standard correctness requirement that for any $(pk, sk) \xleftarrow{R} \mathsf{Kg}(k)$ and message m, if c is an encryption of m under the public key pk, i.e. $c \xleftarrow{R} \mathsf{Enc}_{pk}(m)$, then $\mathsf{Dec}_{sk}(c) = m$.

The standard goal of an encryption scheme is to ensure privacy. Informally, this is the requirement that a ciphertext $\mathsf{Enc}_{pk}(m)$ hides all partial information about m. Notice that this is considerably stronger than asking that a ciphertext does not reveal the entire plaintext, but it is appropriate for the computational framework where messages are not atomic objects, but actual bit-strings. Formalizing this intuition, as well as giving a precise and detailed model of an adversary, is one of the major achievements of the computational community. In the sequel we recall the definitions of two privacy notions for asymmetric encryption schemes, relevant for our results.

INDISTINGUISHABILITY UNDER CHOSEN-PLAINTEXT ATTACK (IND-CPA). Let $\mathcal{AE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme. Our definition of privacy makes use of a *left-right encryption oracle* (or lr-oracle), $\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,b))$, where pk is a public key for encryption (with non-zero probability of being generated by $\mathsf{Kg}(\cdot)$) and b is an internal *selection bit*. The behavior of the oracle is specified in Figure 1. A query to the oracle is a pair (m_0, m_1) of equal-length messages, and the corresponding answer is an encryption of m_b, i.e. $\mathsf{Enc}_{pk}(m_b)$.

Intuitively, the encryption scheme is secure if an adversary that has access to an lr-oracle with the internal selection bit chosen at random, can not determine what this bit is, with probability significantly better over guessing it. Formally, we consider the experiment $\mathbf{Exp}^{\text{ind-cpa\_}b}_{\mathcal{AE},\mathcal{A}}(k)$, described in Figure 1, for a fixed bit b and adversary $\mathcal{A}$. The experiment is as follows. First a pair of public secret keys for encryption scheme $\mathcal{AE}$ is generated by running the key generation algorithm on the security parameter. The public key pk is passed as input to the adversary $\mathcal{A}$ and it is also used to key the lr-oracle. The adversary is given access to the oracle and after interacting with the oracle for as long as it wants, the adversary outputs a guess bit d. One can imagine that adversary $\mathcal{A}$ is in one of the two possible worlds (in which the oracle encrypts the first or the second message of the queries that it receives) and its goal is to determine in which world it is. The adversary is successful if it can do so with non-negligible probability. Formally, we define the advantage of adversary $\mathcal{A}$ as a function of the security parameter by:

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{ind-cpa\_1}}_{\mathcal{AE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cpa\_0}}_{\mathcal{AE},\mathcal{A}}(k) = 1\right]$$

and say that encryption scheme $\mathcal{AE}$ is IND-CPA secure if the function $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\mathcal{A}}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

We illustrate this security definition via a simple example. The example shows that if for an encryption scheme $\mathcal{AE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ from any given ciphertext c, it is always possible to compute some partial information $f(m)$ on the underlying plaintext m, then the scheme is not IND-CPA secure in the sense of the above definition. For this, consider the IND-CPA adversary $\mathcal{A}$ that chooses two equal-length messages m_0 and m_1 such that $f(m_0) \neq f(m_1)$ and submits (m_0, m_1) to the left-right oracle. Next, from the ciphertext returned by the lr-oracle, $\mathcal{A}$ computes $i = f(m_b)$. If $i = f(m_1)$ the adversary returns 1, else it returns 0. Then for any security parameter k, $\Pr\left[\mathbf{Exp}^{\text{ind-cpa\_}\mathcal{A}}_{\mathcal{AE},1}(k) = 1\right] = 1$ and $\Pr\left[\mathbf{Exp}^{\text{ind-cpa\_}\mathcal{A}}_{\mathcal{AE},0}(k) = 1\right] = 0$. It thus follows that the advantage of the adversary $\mathcal{A}$ is 1, i.e. according to the definition, $\mathcal{AE}$ is not IND-CPA secure.

INDISTINGUISHABILITY UNDER CHOSEN-CIPHERTEXT ATTACK (IND-CCA). The strongest privacy notion considered in the computational community, is *indistinguishability under chosen-ciphertext attack*. The adversarial model is considerably strengthened (over the one for IND-CPA), by allowing adversary $\mathcal{A}$ to obtain decryptions of ciphertexts of its choice (hence the name). The formal definition uses the experiment $\mathbf{Exp}^{\text{ind-cca\_}b}_{\mathcal{AE},\mathcal{A}}(k)$ in Figure 1, defined for a fixed bit b, and adversary $\mathcal{A}$. The experiment is similar to the one for defining IND-CPA security, but here, besides access to the lr-encryption oracle, adversary $\mathcal{A}$ is also equipped with a decryption oracle. The goal of the adversary remains the same, i.e. determine the internal selection bit of the lr-oracle.

We prohibit the trivial attack in which the adversary submits two different plaintexts m_0, m_1 to the encryption oracle, and after obtaining the reply c, submits c to the decryption oracle. We thus require that the adversary $\mathcal{A}$ does not query the decryption oracle on a ciphertext which was previously output by the lr-oracle. We now define the advantage of adversary $\mathcal{A}$ in defeating IND-CCA security of encryption scheme $\mathcal{AE}$ by

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},\mathcal{A}}(k) = 1\right]$$

and say that encryption scheme $\mathcal{AE}$ is IND-CCA secure if the function $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{A}}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Oracle $\mathsf{Enc}_{pk}(\mathrm{LR}(m_0, m_1, b))$	Experiment $\mathbf{Exp}^{\text{ind-cpa\_}b}_{\mathcal{AE},\mathcal{A}}(k)$	Experiment $\mathbf{Exp}^{\text{ind-cca\_}b}_{\mathcal{AE},\mathcal{A}}(k)$
$c \xleftarrow{R} \mathsf{Enc}_{pk}(m_b)$	$(pk, sk) \xleftarrow{R} \mathsf{Kg}(k)$	$(pk, sk) \xleftarrow{R} \mathsf{Kg}(k)$
return c	$d \leftarrow \mathcal{A}^{(\mathsf{Enc}_{pk}\mathrm{LR}(\cdot,\cdot,b))}(pk)$	$d \leftarrow \mathcal{A}^{(\mathsf{Enc}_{pk}\mathrm{LR}(\cdot,\cdot,b)),\mathsf{Dec}_{sk}}(pk)$
	return d	return d

Figure 1. The left algorithm defines the way queries are answered by a left-right oracle. The next two experiments are used in defining IND-CPA and IND-CCA security. For the $\mathbf{Exp}^{\text{ind-cca\_}b}_{\mathcal{AE},\mathcal{A}}(k)$ experiment we require that $\mathcal{A}$ does not query to the decryption oracle a ciphertext obtained as a result of a query to the left-right oracle

3 Execution Model

In this section we give a detailed description of the framework that we use to analyze the security of the Needham-Schröeder protocol. The framework is a straightforward adaptation of the framework of [5] to the public key setting.

TWO PARTY PROTOCOLS. A protocol is a set of prescribed behaviors for *principals* (protocol participants) that typically involve local computation and message exchange over a (potentially insecure) network. Principals identities come from a fixed set $\mathsf{IDs} = \{1, 2, ..., N_p\}$, where N_p is assumed to be some fixed constant. We will use variables $A, B, ...$ to denote members of IDs.

We are interested in two party protocols, and make the usual distinction between *initiator* (the principal that sends the first message) and *responder* (the principal for which the first message is intended). The protocols we consider are in the public key setting, and we use standard notation for keys: for each principal A we denote its public and secret keys by pk_A and sk_A, respectively.

GRAPHICAL REPRESENTATION. A common way to represent two party protocols is as in Figure 2 where we give the Needham-Schröeder protocol [17]. Here parties A (as initiator) and B (as responder) interact as follows: A generates a random nonce N_A and sends to B the concatenation of A and N_A encrypted with the public key of B. Upon receiving this message, the responder generates a new random nonce N_B, and sends to A the concatenation of N_A and N_B encrypted under the public key of A. Finally, the initiator encrypts N_B under B's public key and sends it to B. It is implicit that whenever a party observes a deviation from the protocol it aborts, otherwise, if it reaches the end of the protocol, the run is considered successful.
The graphical representation is rather abstract, and the semantics of the symbols appearing in the picture is not fully specified. Of course, K_A, K_B are symbols intended to represent the public keys of A and B respectively, and N_A, N_B represent nonces, but the precise nonce and key spaces are not specified. In contrast, the computational approach gives a very precise treatment of all these details.

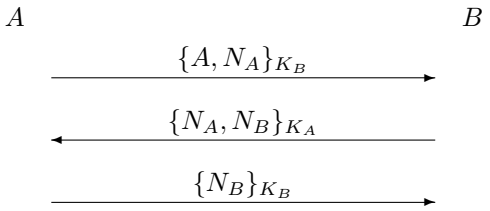

Figure 2. The Needham-Schröeder public key protocol

COMPUTATIONAL APPROACH. We will use the following setting. Protocol participants are modeled as interactive Turing machines computing some probabilistic functions. A two party protocol in the public key setting is defined by a triple of algorithms $\Pi = (\mathcal{G}, \Pi_i, \Pi_r)$. Here, $\mathcal{G}$ is a randomized key generator algorithm, which on input a security parameter k, returns a pair of public/secret key pair (pk_A, sk_A) for each $A \in \mathsf{IDs}$. Algorithms Π_i and Π_r are the programs run by the initiator and responder respectively. Both algorithms take as input the following parameters:

- 1^k - security parameter (in unary representation), $k \in \mathsf{N}$;
- $A \in \mathsf{IDs}$ - the identity of the party running the algorithm;
- $B \in \mathsf{IDs}$ - the identity of the (intended) partner;
- $KI_{A,B}$ - the key information that is available to party A, i.e. pk_A, sk_A, pk_B (note that A does not know the secret key of B);
- tr - the transcript of the conversation so far, i.e. a representation of the list of messages transmitted and received by A in the current run of the protocol;
- R - polynomially many random coin flips (in the security parameter);

For $x \in \{i, r\}$, the output of the function $\Pi_x(1^k, A, B, KI_{A,B}, \mathsf{tr}, R) = (m, \delta)$ specifies:

- $m \in \{0,1\}^* \cup \{*\}$ - the message that is to be sent from A to B; $*$ indicates that no message is to be sent;
- $\delta \in \{\mathsf{A}, \mathsf{R}, *\}$ - the decision taken by party A; it is one of accept, reject, or no decision ;

COMMUNICATION MODEL. The communication model that we use follows the one of [5]. We make the strong, but standard assumption, that the adversary has total control over the network. The adversary obtains messages from the parties, may apply to the them any polynomial time computable function, and then, choose to deliver the resulting messages to the intended recipient, other protocol participants, or drop it from the network altogether. Although our model takes into account the possibility that parties may run multiple instances of the protocol at the same time, we limit their number to some constant N_s. In the rest of the paper we will denote by SNo the set $\{1, 2, ..., N_s\}$ of session ids. One last feature that we point out before going into the details, is that the model also captures breakins.

The adversary that we consider is a probabilistic polynomial-time algorithm $\mathcal{A}$ with access to a finite number of oracles

$$\{\Pi_x^{AB,t} : x \in \{i, r\},\ A, B \in \mathsf{IDs},\ t \in \mathsf{SNo}\}$$

Here, each oracle models *one* session of the protocol run by some party. Concretely, oracle $\Pi_i^{AB,t}$ maintains the state of t'th session run by party A as initiator, with B as intended responder. Similarly, $\Pi_r^{AB,s}$ maintains the state and the information that principal A has while running the s-th session of the protocol as a responder. For example, an initiator oracle would compute the first message of the the protocol, deliver it to the adversary and wait for a valid response. Once a response is obtained, the oracle computes the next message to be sent, delivers it to the adversary and so on.

As we have already anticipated, we put the adversary in charge of message delivery. We assume that the adversary has a special *query tape*, on which it writes its queries to the oracles. Passing/receiving messages from the oracles, is done by using Send queries. More precisely, when the adversary writes $\Pi_i^{AB,s} : \mathsf{Send}(m)$, on the query tape, the oracle $\Pi_i^{AB.s}$ reads the message m; if the message is a valid protocol message, i.e. it is a message A expects to receive from B, the oracle computes the reply M that A would normally output in answer to m, returns M to the adversary and updates its internal state. If message m is invalid, the oracle stops. We write $M \stackrel{R}{\leftarrow} \mathcal{O} : \mathsf{Send}(m)$ for the process of passing message m to oracle O, and obtaining M as a reply. In particular, $M \stackrel{R}{\leftarrow} \mathcal{O} : \mathsf{Send}()$ denotes the execution of the first query to an initiator oracle.

The second type of queries that we consider are $\mathsf{Corrupt}$ queries. These model the possibility that before or during the running the protocol, the adversary compromises one or more protocol participants. We emphasize that $\mathsf{Corrupt}$ queries are not oracle specific, but party specific: when a query $\mathsf{Corrupt}(A)$ is written on the query tape, *all* oracles $\Pi_x^{AI,s}$, $I \in \mathsf{IDs}$ reveal their internal states, i.e. all transcripts of sessions that A is involved in together with the random coins that they are using. In particular the adversary learns the secret key of A. For our purposes it is sufficient that the adversary learns the secret key of the corrupted party. We will write $sk_A \leftarrow \mathsf{Corrupt}(A)$ for the process of corrupting party A. We point out that the adversaries in the model are *adaptive*, i.e. the choice of parties to be corrupted may depend on the messages exchanged with the oracles.

The precise formalization of how oracles handle the queries is given in Figure 3.

We can now give a detailed description of an experiment that we denote by $\mathbf{Exp}_{\Pi,\mathcal{A}}(k)$ intended to model the execution of the protocol Π in the presence of an adversary $\mathcal{A}$ having the capabilities described above. For a fixed security parameter k, the experiment is as follows:

1. For each $A \in \mathsf{IDs}$, pick random R_A of appropriate length and run the key generation algorithm to obtain public-secret keys for party A, i.e. execute $(pk_A, sk_A) \stackrel{R}{\leftarrow} \mathcal{G}(1^k)$.
2. For $x \in \{i, r\}$, for each $A, B \in \mathsf{IDs}$, $A \neq B$ and each $t \in \mathsf{SNo}$, pick random $R_x^{AB,t}$ of appropriate length (this is the randomness used by party A for running the t-th instance of the Π_x protocol with B as intended partner);
3. For $x \in \{i, r\}$, for each $A, B \in \mathsf{IDs}$, each $t \in \mathsf{SNo}$, set $\mathsf{tr}_x^{AB,t} = \emptyset$ (this is the variable that will keep track of the conversation of $\Pi_x^{AB,t}$);
4. Pick random string $R_{\mathcal{A}}$ of appropriate length (this is the randomness used by the adversary);
5. Run adversary $\mathcal{A}$ until it stops, answering its queries as described in Figure 3.

So far we have described how to model the execution of a (two-party) protocol over a network under adversarial control. In the next section we give an example of a specific goal of such protocols and show how to define security of protocols trying to achieve it.

4 Mutual Authentication

The functionality that we are interested in is *mutual authentication*. The goal of mutual authentication protocols, is to ensure that at the end of a successful execution of the protocol between two honest (uncorrupted) parties A and B, both parties are convinced that they "talk" to each other. Various approaches to modeling this property in the computational setting exist [5, 6, 3]. For our result we choose the formalization of [5], based on *matching conversations*, adapted to the public key case. Informally, this is the requirement that following a successful run of the protocol between principals A and B, the two parties that are involved have the same view on the messages that have been exchanged: if party A sent M_1, received M_2 and then sent

Query	Oracle reply	Oracle update
$\Pi_x^{AB,t}$: $\mathsf{Send}(m)$	$(m_o, \delta) = \Pi_x(1^k, A, B, KI_{A,B}, \mathsf{tr}_x^{AB,t}.m_i, R_x^{AB,t})$	$\mathsf{tr}_x^{AB,t} \leftarrow \mathsf{tr}_x^{AB,t}.(m_i, m_o)$
$\mathsf{Corrupt}(A)$	$\langle sk_A, r_x^{AB,t}, \mathsf{tr}_x^{AB,t}, R_x^{AB,t}\rangle_{(B,t,x)\in \mathsf{IDs}\times\mathsf{SNo}\times\{i,r\}}$	

Figure 3. How oracle $\Pi_x^{AB,t}$ handles queries of adversary $\mathcal{A}$

M_3, then party B received M_1 and sent M_2. We stress that it is not required for the last message of the protocol to be delivered, since the adversary can simply drop it from the network. However, if B receives M_3, then A sent M_3 to B. The following formal definitions are from [5].

Definition 4.1 Fix an execution of an adversary $\mathcal{A}$ with access to oracles as we have described above. For each oracle, we define its *conversation* as the sequence

$$C = (\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), ..., (\tau_m, \alpha_m, \beta_m)$$

with the meaning that at time τ_i, the the oracle received as query α_i and answered with β_i for all $1 \leq i \leq m$. We are not interested in concrete values for the τ parameters; we use these parameters to obtain a total order on the messages that are sent during one execution. of the protocol.

Definition 4.2 Consider a two-party protocol $(\Pi_i, \Pi_r, \mathcal{G})$ running in $R = 2\rho - 1$ rounds (the case of even number round protocol is similar); also assume that the last message of the protocol is sent by the initiator. Consider the execution described above in the presence of adversary $\mathcal{A}$. Let C_i and C_r be the conversations of two oracles $\Pi_i^{AB,s}$ and $\Pi_r^{BA,t}$ during this execution.

1. We say that C_r is a matching conversation for C_i if there exists $\tau_0 < \tau_1 < ... < \tau_R$ and $\alpha_1, \beta_1, ..., \alpha_\rho, \beta_\rho$ such that C_i is prefixed by:

   $$(\tau_0, , \alpha_1), (\tau_2, \beta_1, \alpha_2)...$$
   $$..., (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

   and C_r is prefixed by:

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), ..., (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1})$$

2. We say that C_i is a matching conversation for C_r if there exist $\tau_0, \tau_1, ..., \tau_R$ and $\alpha_1, \beta_1, ..., \alpha_R, \beta_R$ such that C_r is prefixed by:

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), ...,$$
   $$..., (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-2}, \alpha_\rho, *)$$

   and C_i is prefixed by

   $$(\tau_0, , \alpha_1), (\tau_2, \beta_1, \alpha_2), ...,$$
   $$..., (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

The above requirements are somewhat asymmetric. The first one says that if an initiator oracle sent messages $\alpha_1, \alpha_2..., \alpha_\rho$, and received $\beta_1, \beta_2, ..., \beta_{\rho-1}$ a responder oracle has a matching conversation if it receives $\alpha_1, ..., \alpha_{\rho-1}$ and sends $\beta_1, ..., \beta_{\rho-1}$; it is not enforced that the responder receives the last message sent by the initiator. In contrast, an initiator has a matching conversation with a responder if *all* messages sent by the responder are received by the initiator.

We will say that oracles $\Pi_x^{AB,s}$ and $\Pi_y^{CD,t}$, with $(x, y) \in \{(i, r), (r, i)\}$, have a matching conversation if the first has conversation C_i, the second has conversation C_r and the two conversation are matching (as defined above).

If during the execution, $\mathcal{A}$ makes a $\mathsf{Corrupt}(A)$ query, we will say that A is a corrupt party.

Mutual authentication can now be defined by requiring that no oracle accepts, without having a corresponding oracle with a matching conversation.

Definition 4.3 Let Π be a two party protocol and let $\mathcal{A}$ be an adversary against Π. Let $\mathsf{Nomatching}^{\mathbf{Exp}_{\Pi,\mathcal{A}}}(k)$ denote the event that after executing experiment $\mathbf{Exp}_{\Pi,\mathcal{A}}(k)$ there exists uncorrupted parties $A, B \in \mathsf{IDs}$, a session number $s \in \mathsf{SNo}$ and $x \in \{i, r\}$ such that oracle $\Pi_x^{AB,s}$ accepts, and there exists no oracle $\Pi_y^{BA,t}$ with $t \in \mathsf{SNo}$ and y (with $(x, y) \in \{(i, r), (r, i)\}$, such that $\Pi_x^{AB,s}$ and $\Pi_y^{BA,t}$ have matching conversations. We define the advantage oft $\mathcal{A}$ running against mutual authentication protocol Π by

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathrm{nom}}(k) = \Pr\left[\mathsf{Nomatching}^{\mathbf{Exp}_{\Pi,\mathcal{A}}}(k)\right].$$

We say that protocol $\Pi = (\mathcal{G}, \Pi_i, \Pi_r)$ is a secure mutual authentication protocol if:

1. **Correctness:** for every polynomial time adversary $\mathcal{A}$, in experiment $\mathbf{Exp}_{\Pi,\mathcal{A}}(k)$, if parties $A, B \in \mathsf{IDs}$ are not corrupt, and oracles $\Pi_i^{AB,s}$ and $\Pi_r^{BA,t}$ have matching conversations then both oracles accept;
2. **Security:** for every polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$, $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathrm{nom}}(\cdot)$ is a negligible function (of the security parameter).

5 Security Analysis of the Needham-Schröeder(-Lowe) Protocol

In this section we analyze the security of the Needham-Schröeder protocol . We start by fixing some imple-

mentation details, and then we move on with the analysis. First, we cast the well-known attack of Lowe [13] to our complexity-theoretic framework. Then, we show that Lowe's variant of the protocol may not be secure if the encryption scheme that is used to implement the protocol is only IND-CPA secure. Finally, we prove that if the encryption scheme is strong enough (IND-CCA secure), then the Needham-Schröeder-Lowe protocol is indeed secure.

CONCRETE IMPLEMENTATIONS. For our analysis, we need to fix some notation and make certain assumptions on the implementation of the protocol. First, we assume that in any implementation of the protocol with security parameter k, the nonces are drawn independently and uniformly at random from the set $\{0,1\}^k$. Thus, we assume that for security parameter k, protocol messages belong to the set:

$$\mathsf{Msg}(\mathsf{k}) = \mathsf{first}(k) \cup \mathsf{second}(k) \cup \mathsf{third}(k)$$

where sets $\mathsf{first}(k) = \mathsf{IDs} \times \{0,1\}^k, \mathsf{second}(k) = \{0,1\}^k \times \{0,1\}^k$ and $\mathsf{third}(k) = \{0,1\}^k$ correspond to the first, second respectively third message of the protocol. For Lowe's version of the protocol, the set $\mathsf{second}(k)$ is modified accordingly. We will denote nonces by $n_A, n_B, ...$, possibly with natural number superscripts. Finally, for any asymmetric encryption scheme $\mathcal{AE}$ for which the set of valid plaintexts includes Msg, we denote by $\mathsf{NS}[\mathcal{AE}]$ and $\mathsf{NSL}[\mathcal{AE}]$ the implementations of the Needham-Schröeder and Needham-Schröeder-Lowe protocols using $\mathcal{AE}$ to encrypt the messages that are exchanged.

A FAMOUS ATTACK ON THE NEEDHAM-SCHRÖEDER PROTOCOL. One of the most famous attacks against a cryptographic protocol is the one given by Lowe against the Needham-Schroeder protocol, [13]. Since the attack is independent of the encryption scheme used to implement the protocol we prove the following theorem by providing the analogous attack in our framework.

Theorem 5.1 *For any asymmetric encryption scheme $\mathcal{AE}$, $\mathsf{NS}[\mathcal{AE}]$ is not a secure mutual authentication protocol.*

Proof: We prove the theorem by translating Lowe's attack to our framework. Since only three parties, say A, B, C, each running one session of the protocol are needed to describe the attack, for notational convenience, in the sequel we will drop the session id superscript from our oracle notation. The details of the adversary that we consider, i.e. its local computations and oracle queries, are given in Figure 4.

The attack starts with the corruption of party C, i.e in particular $\mathcal{A}$ obtains sk_C, the secret key of C. Then, the adversary initializes oracle Π_i^{AC}, the oracle that represents party A trying to run the protocol (as initiator) with party C, and

Adversary $\mathcal{A}$
(1) $sk_C \leftarrow \mathsf{Corrupt}(C)$
(2) $c_1 \stackrel{R}{\leftarrow} \Pi_i^{AC}$: $\mathsf{Send}()$
(3) parse $\mathsf{Dec}_{sk_C}(c_1)$ as A, n_A
(4) $c_2 \stackrel{R}{\leftarrow} \mathsf{Enc}_{pk_B}(A, n_A)$
(5) $c_3 \stackrel{R}{\leftarrow} \Pi_r^{BA}$: $\mathsf{Send}(c_2)$
(6) $c_4 \stackrel{R}{\leftarrow} \Pi_i^{AC}$: $\mathsf{Send}(c_3)$
(7) parse $\mathsf{Dec}_{sk_C}(c_4)$ as n_B
(8) $c_5 \stackrel{R}{\leftarrow} \mathsf{Enc}_{pk_B}(n_B)$
(9) $c_6 \stackrel{R}{\leftarrow} \Pi_r^{BA}$: $\mathsf{Send}(c_5)$

Figure 4. The computational version of Lowe's attack on the Needham-Schröeder protocol

obtains in return c_1, the first message sent by A. Here c_1 is the the encryption of the identity A and nonce n_A under the public key of party C. Using the secret key sk_C, the adversary decrypts the ciphertext and obtains n_A. In the next step of the attack, the adversary sends to Π_r^{BA} the encryption of A, n_A under the public key of B. This initializes a new run of the protocol between parties A as initiator and B as responder. Oracle Π_r^{BA} responds with c_3, the encryption of n_A and n_B, a newly generated nonce, under the public key of A. The adversary sends c_3 to oracle Π_i^{AC}; note that although at this point the adversary does not know n_B, c_3 is precisely the type of message that oracle Π_i^{AC} is expecting. The answer of the oracle is c_4, the encryption of n_B under C's public key (which the adversary can decrypt.) Finally, the adversary sends c_5 the encryption of n_B under pk_B to Π_r^{BA}. It is immediate that oracle Π_r^{BA} accepts, although no oracle Π_i^{AB} has a matching conversation, or formally, $\mathbf{Adv}^{\mathrm{nom}}_{\Pi,\mathcal{A}}(k) = 1$ for any security parameter k. We conclude that the Needham-Schröeder protocol is not a secure mutual authentication protocol for any instantiation of the encryption scheme. ▌

A LESS FAMOUS ATTACK ON THE NEEDHAM-SCHRÖEDER-LOWE PROTOCOL. In a subsequent paper ([14]), Lowe suggests and proves secure a modified version of the Needham-Schröeder protocol. The fix that he suggests is to simply replace the second message of the protocol with the message $\{B, N_A, N_B\}_{K_A}$. Various proofs of security, for the resulting protocol (that we will refer to as the Needham-Schröeder-Lowe protocol) have been given since, all in the formal methods framework. Our next result issues a cautionary note with respect to security of protocols proved in this very abstract setting. Namely, we prove the following theorem:

Proposition 5.2 *There exists an IND-CPA secure encryption scheme $\mathcal{AE}$ such that the* $\mathsf{NSL}[\mathcal{AE}]$ *protocol is not a secure mutual authentication protocol.*

Proof: We prove the theorem in three steps. We start by constructing an encryption scheme $\overline{\mathcal{AE}}$ that is IND-CPA secure. Then, we show how to modify $\overline{\mathcal{AE}}$, in order to obtain a new IND-CPA secure encryption scheme $\mathcal{AE}$, suitable for implementing the NSL protocol. In the last step of the proof, we exhibit an adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathrm{nom}}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k)$ is non-negligible.

Let $(p_k)_{k\in\mathsf{N}}$ be a sequence of Sophie-Germaine primes (i.e. $q_k = 2p_k + 1$ is also prime for all k), for which $|p_k| = 3k$. We define $\overline{\mathcal{AE}} = (\overline{\mathsf{Kg}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ in Figure 5. Under the widely accepted Decisional Diffie-Helman assumption, the ElGamal encryption scheme $\mathcal{AE}$ defined above is IND-CPA secure, [8]. However, the set of valid plaintexts of $\overline{\mathcal{AE}}$ is $\mathsf{QR}(q_k)$ the group of quadratic residues modulo q_k, i.e. $\mathsf{QR}(q_k) = \{x^2 \bmod q_k : x \in \mathsf{N}\}$, so the scheme can not be used directly in the implementation of the protocol.

In order to transform $\overline{\mathcal{AE}}$ into an encryption scheme suitable for implementing the NSL protocol, we postulate the existence of a sequence of invertible encoding functions $\langle\cdot\rangle_k : \mathsf{Msg}(k) \to \mathsf{QR}(q_k)$. We will denote by $\langle\cdot\rangle_k^{-1}$ the corresponding inverse mappings. The new encryption scheme $\mathcal{AE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

1. $\mathsf{Kg} = \overline{\mathsf{Kg}}$, i.e. the key generation algorithm stays the same;
2. $\mathsf{Enc}_{pk}(m) = \overline{\mathsf{Enc}}_{pk}(\langle m\rangle_k)$, i.e. to encrypt a message m, first the encoding function is applied to m and the resulting element of $\mathsf{QR}(q_k)$ is encrypted using the encryption algorithm of $\overline{\mathcal{AE}}$.
3. $\mathsf{Dec}_{sk}(c) = \langle\overline{\mathsf{Dec}}_{sk}(c)\rangle_k^{-1}$ i.e. decryption is the reverse process, in which a ciphertext is first decrypted using the decryption algorithm of $\overline{\mathcal{AE}}$ and then the inverse of the encoding function is applied to obtain the original message.

We will further assume that the encoding functions satisfies the following algebraic relation: for some $B, C \in \mathsf{IDs}$ there exists some $x \in \mathsf{QR}(q_k)$ such that $x \cdot \langle B, n_0, n_1\rangle_k \equiv \langle C, n_0, n_1\rangle_k \pmod{q_k}$ for all $n_0, n_1 \in \{0,1\}^k$. Although $\mathcal{AE}$ has the undesirable property that $\mathsf{Enc}_{pk}(\langle C, n_0, n_1\rangle_k) = x \cdot \mathsf{Enc}_{pk}(\langle B, n_0, n_1\rangle_k)$, it is possible to show that the encryption scheme $\mathcal{AE}$ is IND-CPA secure.

Using the above property, we can now modify Lowe's attack on the original protocol (Figure 4) in order to obtain an attack on the $\mathsf{NSL}[\mathcal{AE}]$. Simply observe that if c_3 is an encryption of (B, n_A, n_B) under pk_A, then $x \cdot c_3$ is an encryption of (C, n_A, n_B). We thus replace line (6) of the adversary in Figure 4 and the attack succeeds as before. ∎

SECURITY OF THE NEEDHAM-SCHRÖEDER-LOWE PROTOCOL. We now identify a condition on the encryption scheme, sufficient to ensure the security of the protocol. Our result is captured by the following theorem.

Theorem 5.3 *For any IND-CCA secure asymmetric encryption scheme $\mathcal{AE}$,* $\mathsf{NSL}[\mathcal{AE}]$ *is a secure mutual authentication protocol.*

Proof: (sketch)

Correctness of the protocol is immediate: oracles with matching conversations always accept. The interesting part of the proof is security, which we prove by standard reduction arguments. We show that if there exists an adversary $\mathcal{A}$ that defeats the $\mathsf{NSL}[\mathcal{AE}]$ protocol then there exists an adversary $\mathcal{B}$ breaking the encryption scheme.

We first introduce some notation. We will denote by Index the set $\{(A, B, s, x) \in \mathsf{IDs} \times \mathsf{IDs} \times \mathsf{SNo} \times \{i, r\} : A \neq B\}$. If $\mathcal{O}$ denotes some oracle, say $\Pi_x^{AB,s}$, we denote by $\mathsf{E}_{\mathcal{O}}$ the event that after executing experiment $\mathbf{Exp}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k)$ oracle $\mathcal{O}$ accepts, but no oracle $\Pi_y^{BA,t}$ (with $(x, y) \in \{(i, r), (r, i)\}$) has a matching conversation with $\mathcal{O}$. Although the event depends on the security parameter, for notational convenience we omit to explicitly show this dependence. Using the definition of the advantage of $\mathcal{A}$, we obtain that :

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{nom}}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k) = \\
&\quad = \Pr\left[\mathsf{Nomatching}^{\mathbf{Exp}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k)}\right] \\
&\quad = \Pr\left[\exists (A, B, s, x) \in \mathsf{Index} : \mathsf{E}_{\Pi_x^{AB,s}}\right] \\
&\quad \leq \sum_{A,B,x,s} \Pr\left[\mathsf{E}_{\Pi_x^{AB,s}}\right]
\end{aligned}$$

By the assumption that $\mathsf{NSL}[\mathcal{AE}]$ is insecure, there exists an adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathrm{nom}}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(\cdot)$ is a non-negligible function. Therefore, there exists a polynomial $p(\cdot)$ such that $\mathbf{Adv}^{\mathrm{nom}}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k) > 1/p(k)$, for infinitely many security parameters k. By a simple averaging argument, there must exist $(A, B, x, s) \in \mathsf{Index}$ such that

$$\Pr\left[\mathsf{E}_{\mathcal{O}}\right] \geq \frac{1}{p(k) \cdot |\mathsf{Index}|} \tag{1}$$

for infinitely many security parameters k. Since we have assumed that both the number of parties and the number of sessions are constants, the expression on the right of the inequality is a non-negligible function, and therefore so is the one on the left.

Up to this point we have shown that there exists an adversary $\mathcal{A}$ and an oracle $\mathcal{O}$, such that with non-negligible probability, $\mathcal{A}$ causes $\mathcal{O}$ to accept without having a matching

Algorithm $\mathsf{Kg}(k)$
$r \xleftarrow{R} \mathsf{Z}_{p_k}$; $g \leftarrow r^2$
$x \xleftarrow{R} \mathsf{Z}_{p_k}$
$sk \leftarrow x$; $pk \leftarrow (g, g^x)$
output (sk, pk)

Algorithm $\mathsf{Enc}_{(g,z)}(m)$
$r \xleftarrow{R} \mathsf{Z}_{p_k}$
$y \leftarrow z^r$
$C \leftarrow (g^r, y \cdot m)$

Algorithm $\mathsf{Dec}_x((t, w))$
$m \leftarrow w/t^x$
output m

Figure 5. An ElGamal encryption scheme

conversation with an appropriate oracle. In the sequel we will show how to use $\mathcal{A}$ to build an adversary that is successful against the encryption scheme. In doing so, we distinguish between the cases when $\mathcal{O}$ is an initiator oracle or a responder oracle.

CASE 1. We assume for now that $\mathcal{O}$ is an initiator oracle, i.e. $\mathcal{O} = \Pi_i^{AB,s}$ for some $(A, B, s, i) \in \mathsf{Index}$. Let c_1 be the answer returned by $\mathcal{O}$ in response to the query $\mathcal{O} : \mathsf{Send}()$. We will denote by $\mathsf{E_{ask}}$ the event that at a later point, $\mathcal{A}$ makes a query $\mathsf{Send}(c_1)$ to some responder oracle $\Pi_r^{BA,t}$, and by $\overline{\mathsf{E_{ask}}}$ we denote the opposite event.

Let us first assume that event $\mathsf{E_{ask}}$ does not occur. If c_1 is the encryption of A, n_A, then at a later point $\mathcal{A}$ must make a second query to $\mathcal{O}$ of the type $\mathsf{Enc}_{pk_A}(B, n_A, n_B)$. Intuitively, since c_1 is not submitted to any of the oracles that could possibly decrypt it, it must be the case that somehow the adversary manages to recover n_A from c_1 itself.

This intuition underlies adversary $\mathcal{B}_1$ in Figure 6. The adversary has access to an lr-encryption oracle, $\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot, \cdot, b))$ keyed with some key pk, and to a decryption oracle keyed with the associated key sk. What $\mathcal{B}_1$ does, is to set up an environment in which it can run $\mathcal{A}$: it generates public-secret key pairs for all parties in the system, except for party B; the public key of B is set to be pk the key $\mathcal{B}_1$ obtains from the experiment in which is run, and which $\mathcal{B}_1$ tries to break. Then, $\mathcal{B}_1$ executes steps 2,3,4 of the experiment $\mathbf{Exp}_{\mathsf{NSL}[\mathcal{AE}],\mathcal{A}}(k)$ and starts running $\mathcal{A}$ in this environment.

The simulation of the environment can be carried out perfectly, since $\mathcal{B}_1$ can answer any query that $\mathcal{A}$ makes. We give a detailed description of how the queries are answered, and keep in mind that this is how all adversaries that we construct proceed in a similar method:

- $\mathsf{Corrupt}(P)$: if P is different than B, then $\mathcal{B}_1$ simply returns sk_P to $\mathcal{A}$ (which it knows since it generated it by itself)
- $\mathsf{Corrupt}(B)$: $\mathcal{A}$ does not know the secret key of B, but we since we assume that A, B are honest participants this query does not happen. If it does, we simply abort.
- $\Pi_x^{CD,t} : \mathsf{Send}(c)$: if $C \neq B$ and $\Pi_x^{CD,t} \neq \mathcal{O}$, then $\mathcal{B}_1$ decrypts c using sk_C, and then answers following the protocol.
- $\mathsf{Send}(c) : \Pi_x^{BD,t}$: adversary $\mathcal{A}$ obtains the decryption of c from the decryption oracle, and then answers by following the protocol
- queries to $\mathcal{O}$ vary from adversary to adversary; handling of queries made by $\mathcal{B}_1$ is described bellow.

Adversary $\mathcal{B}_1^{\mathsf{Enc}_{pk}(\mathrm{LR}(b,,,)\mathsf{Dec}_{sk}(\cdot)}(pk)$
(1) For $P \in \mathsf{IDs} - \{B\}$ do $(pk_P, sk_P) \xleftarrow{R} \mathcal{G}(1^k)$;
(2) $pk_B \leftarrow pk$;
(3) For $(I, J, t, x) \in \mathsf{Index}$
(4) initialize oracles $\Pi_x^{IJ,t}$;
(5) Run adversary $\mathcal{A}$:
(6) - decrypt queries to oracles $\Pi_x^{IJ} \neq \mathcal{O}$
using the secret key of I
or the decryption oracle
answer the queries as in Figure 3
(7) - answer the query $\mathcal{O} : \mathsf{Send}()$ as follows
(8) $n_A^0, n_A^1 \xleftarrow{R} \{0,1\}^k$ (assume $n_A^0 \neq n_A^1$)
(9) $c \xleftarrow{R} \mathsf{Enc}_{pk}(\mathrm{LR}((A, n_A^0), (A, n_A^1), b))$
(10) return c to $\mathcal{A}$
(11) - when it makes the query $\mathcal{O} : \mathsf{Send}(C)$
(12) parse $\mathsf{Dec}_{sk}(C)$ as (B, n_A, n_B)
(13) if $n_A = n_A^1$ then $d \leftarrow 1$
(14) if $n_A = n_A^0$ then $d \leftarrow 0$
(15) else $d \xleftarrow{R} \{0,1\}$;
(16) return d

Figure 6. Construction of adversary $\mathcal{B}_1$ against $\mathcal{AE}$ starting from an adversary $\mathcal{A}$ against $\mathsf{NSL}[\mathcal{AE}]$

Queries to oracle $\mathcal{O}$ are treated differently: when $\mathcal{A}$ issues its first query to $\mathcal{O}$, i.e. it writes $\mathcal{O} : \mathsf{Send}()$ on its query tape, $\mathcal{B}_1$ randomly selects two nonces n_A^0 and n_A^1 and submits to the lr-oracle the pair of messages $((A, n_A^0), (A, n_A^1))$. The ciphertext that it obtains, i.e. the encryption of (A, n_A^b) under pk is forward to $\mathcal{A}$ as answer to its query. The hope is that since $\mathcal{A}$ manages to make $\mathcal{O}$ accept, its next query to $\mathcal{O}$ will be a ciphertext that is the

encryption of (B, n_A^b, n_B) under the public key of party A. If this is the case, then $\mathcal{B}_1$ can determine b by decrypting this ciphertext and comparing n_A with n_A^0, n_A^1. So, when $\mathcal{A}$ makes its query $\mathcal{O} : \mathsf{Send}(C)$, the adversary $\mathcal{B}_1$ decrypts C using the public key of A, and parses the plaintext as B, n_A, n_B. If n_A is equal to n_A^b (for either $b = 0$ or 1) then $\mathcal{B}_1$ outputs b, else it outputs a randomly chosen bit.

It is important to observe that since we assume that event $\mathsf{E}_{\mathsf{ask}}$ does not occur, the ciphertext c obtained from the lr-encryption oracle is never queried to an oracle $\Pi_r^{BA,t}$, so it is never submitted to the decryption oracle, so $\mathcal{B}_1$ is a valid CCA adversary.

The analysis of the behavior of $\mathcal{B}_1$ is based on the following observation. Whenever adversary $\mathcal{A}$ makes oracle $\mathcal{O}$ accept without querying c to a responder oracle $\Pi_r^{BA,t}$, i.e. there is no responder oracle having a matching conversation with $\mathcal{O}$, adversary $\mathcal{B}_1$ correctly recovers n_B^b, and thus correctly determines the selection bit. If nonce n_B^b is not recovered correctly, the value recovered is equal to $n_B^{\bar{b}}$ with probability at most $1/2^k$ (since $n_B^{\bar{b}}$ was chosen at random). In this case $\mathcal{B}_1$ returns the wrong answer. Finally, if the recovered nonce is different from both n_A^0, n_A^1 (event that happens with probability $1 - \frac{1}{2^k} - \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]$), then $\mathcal{B}_1$ outputs the right answer with probability $1/2$. Formally, for each selection bit $b = 0, 1$ we have the relation:

$$\begin{aligned}
&\Pr\left[\,\mathbf{Exp}^{\text{ind-cca_}b}_{\mathcal{AE},\mathcal{B}_1}(k) = b\,\right] = \\
&= \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right] + \frac{1}{2}\left(1 - \frac{1}{2^k} - \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]\right) \\
&= \frac{1}{2} + \frac{1}{2}\Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}})\,\right] - \frac{1}{2^{k+1}} \qquad (2)
\end{aligned}$$

which can be used to compute the advantage of $\mathcal{B}_1$:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_1}(k) &= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_1}(k) = 1\,\right] - \\
&\quad \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},\mathcal{B}_1}(k) = 1\,\right] \\
&= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_1}(k) = 1\,\right] + \\
&\quad \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},\mathcal{B}_1}(k) = 0\,\right] - 1 \\
&= \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right] - \frac{1}{2^k} \qquad (3)
\end{aligned}$$

We restate that it is essential that event $\mathsf{E}_{\mathsf{ask}}$ does not occur: otherwise $\mathcal{B}_1$ is not a valid CCA adversary since it would have to query c_1 (obtained from the lr-oracle) to the decryption oracle. We treat this case separately, by constructing a second adversary which works well if $\mathsf{E}_{\mathsf{ask}}$ occurs.

Let c_1, the encryption of (A, n_A) under pk_B, be the first message output by oracle $\mathcal{O}$. We assume that c_1 is queried to some oracle $\mathcal{O}' = \Pi_r^{BA,t}$ (i.e. we assume $\mathsf{E}_{\mathsf{ask}}$). Let c_2 be the ciphertext returned, i.e. c_2 is the encryption of (B, n_A, n_B) under pk_A (for some nonce n_B). Since $\mathcal{O}$ accepts, the adversary must make a query $\mathcal{O} : \mathsf{Send}(c_2')$, where c_2' is the encryption of B, n_A, n_B'. While it would be possible that $n_B' = n_B$, it is not possible that $c_2 = c_2'$, since in this case $\mathcal{O}$ and $\mathcal{O}'$ would have matching conversations. Therefore, the adversary produces a ciphertext c_2' which is related to the ciphertext c_2 output by $\mathcal{O}'$ in a meaningful and known way (the first part of the ciphertext is the same, i.e. A, n_A). So, in some sense $\mathcal{A}$ breaks the encryption under the public key of A. We now construct an CCA adversary exploiting this property.

The details of adversary $\mathcal{B}_2$ are given Figure 7. It sets up the environment in which to run $\mathcal{A}$. It runs the key generation algorithm to obtain the public keys of all parties except party A. The public key of party A is set to pk. Then adversary chooses two nonces n_A^0, n_A^1 and start simulating the attack of adversary $\mathcal{A}$. As before, queries to oracles different than $\mathcal{O}$ are answered using the secret keys or the decryption oracle, and $\mathsf{Corrupt}(I)$ queries are answered by returning sk_I to $\mathcal{A}$. Queries to $\mathcal{O}$ are treated differently: When $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}()$, $\mathcal{B}_2$ passes as answer the encryption c of (A, n_A^1) under the public key of B. Queries $\Pi_r^{BA,s} : \mathsf{Send}(c)$ are answered as follows. $\mathcal{B}_2$ generates a fresh random nonce n_B, and submits to the oracle the pair (m_0, m_1), where $m_0 = (B, n_A^0, n_B)$ and $m_1 = (B, n_A^1, n_B)$. The ciphertext c_1 returned by the oracle, i.e. the encryption of m_b under the public key of A is passed to $\mathcal{A}$ as answer to its query.

Suppose the selection bit of the lr-oracle is 1. Then, message $c_1 = \mathsf{Enc}_{pk_B}(A, n_A^b)$ is a valid protocol message and the message that $\mathcal{O}$ expects to receive is a ciphertext c_2 that encrypts (B, n_A^b, n_B) under the public key of A. So, in principle $\mathcal{B}_1$ can determine that $b = 1$ by decrypting c_2 (using the decryption oracle) and comparing the nonce encrypted in c_2 with n_A^0 and n_A^1. The requirement that c_2 was not produced by the lr-oracle is satisfied, since otherwise there must exist some responder oracle which has a matching conversation with $\mathcal{O}$. So, when $b = 1$, the adversary $\mathcal{B}_2$ outputs 1 in the IND-CCA experiment when it correctly determines that $n_A = n_A^1$, which happens with probability $\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}$. If $n_A \neq n_A^1$ the output will be 1 with probability $1/2$. Formally,

$$\begin{aligned}
&\Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_2}(k) = 1\,\right] \\
&\geq \Pr\left[\,\mathsf{E}(\mathcal{O}) \wedge \mathsf{E}_{\mathsf{ask}}\,\right] + \frac{1}{2} \cdot \left(1 - \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,\right]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,\right] \qquad (4)
\end{aligned}$$

Let us analyze the behavior of $\mathcal{B}_2$ when the lr-oracle selection bit is 0. In this case the first message that $\mathcal{O}$ outputs is

Adversary $\mathcal{B}_2^{\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,b)),\mathsf{Dec}_{sk}(\cdot)}(pk)$

(1) For $P \in \mathsf{IDs} - \{A\}$ do $(pk_P, sk_P) \xleftarrow{R} \mathcal{G}(1^k)$;
(2) $pk_A \leftarrow pk$;
(3) $n_A^0, n_A^1 \xleftarrow{R} \{0,1\}^k$
(4) For $(I, J, t, x) \in \mathsf{Index}$
(5) initialize oracles $\Pi_x^{I,J,t}$
(6) Run adversary $\mathcal{A}$:
(7) - decrypt queries to oracles $\Pi_x^{I,J,s} \neq \mathcal{O}$
using the secret key of I if $I \neq A$
using the decryption oracle if $I = A$
answer queries as in Figure 3
(8) - answer the query $\mathcal{O} : \mathsf{Send}()$ as follows:
(9) $c \xleftarrow{R} \mathsf{Enc}_{pk_B}(A, n_A^1)$;
(10) return c to $\mathcal{A}$
(11) - answer a query $\Pi_r^{B,A,s} : \mathsf{Send}(c)$ (for all s)
(12) $n_B \xleftarrow{R} \{0,1\}^k$
(13) $c_2 \xleftarrow{R} \mathsf{Enc}_{sk_A}(\mathrm{LR}((B, n_A^0, n_B), (B, n_A^1, n_B), b)$
(14) return c_2 to $\mathcal{A}$
(15) - when $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}(c_3)$
(16) parse $\mathsf{Dec}_{sk_A}(c_3)$ as B, n_A, n_B
(17) if $n_A = n_A^1$ then $d \leftarrow 1$
(18) if $n_A = n_A^0$ then $d \leftarrow 0$
(19) else $d \xleftarrow{R} \{0,1\}$
(20) return d

Adversary $\mathcal{B}_3^{\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,b)),\mathsf{Dec}_{sk}(\cdot)}(pk)$

(1) For $P \in \mathsf{IDs} - \{B\}$ do $(pk_P, sk_P) \xleftarrow{R} \mathcal{G}(1^k)$;
(2) $pk_B \leftarrow pk$;
(3) $n_A^0, n_A^1 \xleftarrow{R} \{0,1\}^k$
(4) For $(I, J, t, x) \in \mathsf{Index}$
(5) initialize oracles $\Pi_x^{I,J,t}$
(6) Run adversary $\mathcal{A}$:
(7) - decrypt queries to oracles $\Pi_x^{I,J,s} \neq \mathcal{O}$
using the secret key of I if $I \neq B$
using the decryption oracle if $I = B$
answer queries as in Figure 3
(9) - answer the query $\mathcal{O} : \mathsf{Send}()$
(10) $c \xleftarrow{R} \mathsf{Enc}_{pk_B}(\mathrm{LR}((A, n_A^0), (A, n_A^1), b))$;
(11) return c to $\mathcal{A}$
(12) - answer the query $\Pi_r^{B,A,s} : \mathsf{Send}(c)$
(13) $n_B \xleftarrow{R} \{0,1\}^k$
(14) $c_2^s \xleftarrow{R} \mathsf{Enc}_{sk_A}((B, n_A^0, n_B)$
(15) return c_2^s to $\mathcal{A}$
(16) - when $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}(c_3)$
(17) parse $\mathsf{Dec}_{sk_A}(c_3)$ as B, n_A, n_B
(18) if $n_A = n_A^0$ then $d \leftarrow 0$
(19) else $d \leftarrow 1$
(20) return d

Figure 7. Construction of adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$ against $\mathcal{AE}$ from an adversary $\mathcal{A}$ against $\mathsf{NSL}[\mathcal{AE}]$

c_1 the encryption of (A, n_A^1), but when c_1 is queried by $\mathcal{A}$ to oracles $\Pi_r^{B,A,s}$, the answer it obtains in return has the pattern $\mathsf{Enc}_{pk_A}(B, n_A^0, n_B)$; we claim that this ne-concordance between protocol messages does not significantly affect the behavior of $\mathcal{A}$ and that the message c_2 that $\mathcal{A}$ will query to $\mathcal{O}$ will have the same pattern as the ones obtained from the responder oracles. Assume for now that this is the case, then $\mathcal{B}_2$ can recover the nonce n_A^1 simply by querying c_2 to the decryption oracle. Since the only ciphertexts obtained from the lr-oracle correspond to answers of the oracles $\Pi_r^{B,A,t}$, it follows that c_2 was not produced using a query to the lr-oracle (otherwise there would be an oracle with a matching conversation with $\mathcal{O}$), and so $\mathcal{B}_2$ can submit c_2 to the decryption oracle.

Let $\mathsf{E}'_{\mathsf{ask}}$ denote the event that the second query that $\mathcal{A}$ makes to $\mathcal{O}$ is the encryption $\mathsf{Enc}_{pk_A}(B, n_A^0, n_B)$ (for some n_B), then in the CCA experiment $\mathcal{B}_2$ determines that selection bit was 0 whenever the event $\mathsf{E}'_{\mathsf{ask}}$ occurs, or with probability $1/2$ if the event does not occur. Thus we have:

$$\begin{aligned}\Pr\left[\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},\mathcal{B}_2}(k) = 0\right] &= \Pr[\mathsf{E}'_{\mathsf{ask}}] + \frac{1}{2}\cdot(1 - \Pr[\mathsf{E}'_{\mathsf{ask}}]) \\ &= \frac{1}{2} + \frac{1}{2}\cdot\Pr[\mathsf{E}'_{\mathsf{ask}}]\end{aligned}$$

and thus

$$\Pr\left[\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},\mathcal{B}_2}(k) = 1\right] = \frac{1}{2} - \frac{1}{2}\cdot\Pr[\mathsf{E}'_{\mathsf{ask}}] \quad (5)$$

We now justify the claim that the behavior of $\mathcal{A}$ remains essentially unmodified if the oracles $\Pi_r^{B,A,t}$ deviate from the protocol as described above (the first message of oracle $\mathcal{O}$ is $c_1 = \mathsf{Enc}_{pk_B}(A, n_A^1)$, but the answer to a query $\Pi_r^{B,A,t} : \mathsf{Send}(c_1)$ is a ciphertext c_2 which encrypts $(A, n_A^0, n_B$ under the public key of A). The intuition is that if adversary $\mathcal{A}$ significantly deviates from its normal behavior, it must be the case that it obtains information about the underlying plaintexts. We construct a new CCA adversary $\mathcal{B}_3$ (Figure 7) which breaks the encryption under the public key of A. Adversary $\mathcal{B}_3$ prepares the environment in which $\mathcal{A}$ can be run; it obtains the keys for all parties involved, except for party A, by running the key generation algorithm of the encryption scheme. The public key of A is set to pk, the public key of the encryption scheme $\mathcal{B}_2$ has to break. After $\mathcal{B}_3$ sets up the oracles, it starts running $\mathcal{A}$. The queries that $\mathcal{A}$ makes (to oracles other than $\mathcal{O}$ and $\Pi_r^{B,A,t}$) are an-

swered using the secret keys of the parties or the decryption oracle as before. To answer query $\mathcal{O} : \mathsf{Send}()$, $\mathcal{B}_3$ obtains the encryption c of one of the two messages $m_0 = (A, n_A^0)$ or $m_1 = (A, n_A^1)$ from the lr-oracle, and returns c to $\mathcal{A}$. However, when the adversary submits queries ciphertext c to some oracle $\Pi_r^{BA,t}$, $\mathcal{B}_3$ answers as if the plaintext associated to c is m_0, i.e. it generates a fresh nonce n_B and returns to $\mathcal{A}$ the encryption of (B, n_A^0, n_B).

If the selection bit is 0, then all oracles respect the protocol, so $\mathcal{B}_3$ recovers the nonce n_A^0 with probability $\Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,]$, and thus returns 1 with probability at most $(1 - \Pr[\,\mathsf{E}(\mathcal{O} \wedge \mathsf{E}_{\mathsf{ask}}\,])$:

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},\mathcal{B}_3}(k) = 1\,\right] \leq 1 - \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,] \quad (6)$$

For the case that the selection bit is 1, the environment of $\mathcal{A}$ is exactly the one of the experiment $\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},\mathcal{B}_2}(k)$, and thus if event $\mathsf{E}'_{\mathsf{ask}}$ occurs, adversary B_3 will output 0. It follows that

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},\mathcal{B}_3}(k) = 1\,\right] = 1 - \Pr[\,\mathsf{E}'_{\mathsf{ask}}\,] \quad (7)$$

Putting together Equations (4) and (5) we obtain:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_2}(k) &= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_2}(k) = 1\,\right] \\
&\quad \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},\mathcal{B}_2}(k) = 1\,\right] \\
&\geq \frac{1}{2} \cdot (\Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,] + \Pr[\,\mathsf{E}'_{\mathsf{ask}}\,]) \\
&= \frac{1}{2} \cdot (\Pr[\,\mathsf{E}'_{\mathsf{ask}}\,] - \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,]) + \\
&\quad \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,]
\end{aligned}$$

By rearranging the terms and using Equations (6) and (7), we obtain

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_2}(k) + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_3}(k) \geq \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,] \quad (8)$$

We can now finalize the first part of the proof by observing that

$$\Pr[\,\mathsf{E}_{\mathcal{O}}\,] = \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E}_{\mathsf{ask}}\,] + \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]$$

Using Equations (1),(3) and (8) we obtain

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_1}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_2}(k) + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_3}(k) \geq \frac{1}{p(k)|\mathsf{Index}|} - \frac{1}{2^k}$$

Since the function on the right side of the inequality is non-negligible, so is at least one of the terms of the summation on the left side, i.e. at least one of the adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ succeeds in breaking IND-CCA security of $\mathcal{AE}$.

CASE 2. We now tackle the case when the "faulty" oracle is a responder oracle $\mathcal{O} = \Pi_r^{AB,s}$ (for some $(A, B, s, r) \in \mathsf{Index}$.) The proof is also by reduction. We show how to construct three adversaries against the encryption scheme $\mathcal{AE}$, such that at least one is successful in an CCA attack.

Let c_1, c_2 and c_3 be the conversation of $\mathcal{O}$, i.e. the first query of adversary $\mathcal{A}$ to $\mathcal{O}$ is $\mathcal{O} : \mathsf{Send}(c_1)$, the answer returned is c_2 and the second and last query of $\mathcal{A}$ to $\mathcal{O}$ is $\mathcal{O} : \mathsf{Send}(c_3)$.

We will denote by

1. $\overline{\mathsf{E}_{\mathsf{ask}}}$ the event that upon receiving c_2, adversary $\mathcal{A}$ does not make a valid query $\Pi_i^{BA,t} : \mathsf{Send}(c_2)$
2. $\mathsf{E}_{\mathsf{sask}}$ the event that upon receiving c_2, adversary $\mathcal{A}$ makes a *single* valid query $\mathcal{O}' : \mathsf{Send}(c_2)$ to exactly one oracle $\mathcal{O}' = \Pi_i^{BA,t}$
3. $\mathsf{E}_{\mathsf{mask}}$ the event that upon receiving c_2, adversary $\mathcal{A}$ submits c_2 to at least two oracles of the type $\Pi_i^{BA,t}$, and these queries are valid.

Observe that event $\mathsf{E}_{\mathsf{mask}}$ can occur only with non-negligible probability: if $\mathcal{A}$ makes more than two valid queries $\mathsf{Send}(c_2)$ to different initiator oracles, the nonces that these two oracles have encrypted in their first message must be the same. Since we assume that A is an honest party, this happens with probability at most $\frac{1}{2^k}$. Formally:

$$\Pr[\,\mathsf{E}_{\mathsf{mask}}\,] \leq \frac{1}{2^k} \quad (9)$$

Lets assume for the time being that event $\overline{\mathsf{E}_{\mathsf{ask}}}$ occurs, i.e. $\mathcal{A}$ does not query c_2 to some oracle $\Pi_i^{BA,t}$. Since oracle $\mathcal{O}$ accepts, upon sending message $c_2 = \mathsf{Enc}_{pk_A}(B, n_A, n_B)$ for some $n_A, n_B \in \{0,1\}^k$, the adversary will make at a later time a query $\mathcal{O} : \mathsf{Send}(\mathsf{Enc}_{pk_A}(n_B))$. Since we assumed that $\mathcal{A}$ does not query message c_2 to any oracle, intuitively $\mathcal{A}$ manages to obtain n_B from ciphertext c_2 itself, i.e. $\mathcal{A}$ breaks the encryption under the public key of A.

The above attack is captured by adversary $\mathcal{B}_4$ that we give in Figure 8.

As before, $\mathcal{B}_4$ sets up an environment in which it can run $\mathcal{A}$. The keys of all parties, except the one of A are obtained by running the key generation algorithm. The public key of A is set to pk, the key passed to $\mathcal{B}_4$ as input. Once keys are assigned to all parties, $\mathcal{B}_4$ starts running adversary $\mathcal{A}$. It answers the queries of $\mathcal{A}$ as before: when it receives a query, it decrypts it (using the secret key of the party to which the query is addressed, or the decryption oracle if the party is A) and answers it according to the protocol. The only queries that are answered differently are the ones to

Adversary $\mathcal{B}_4^{\mathsf{Enc}_{pk}(\mathrm{LR}(b,\cdot,\cdot)),\mathsf{Dec}_{sk}(\cdot)}(pk)$

(1) For $P \in \mathsf{IDs} - \{A\}$ do $(pk_P, sk_P) \xleftarrow{R} \mathcal{G}(1^k)$;
(2) $pk_A \leftarrow pk$;
(3) For $(I, J, t, x) \in \mathsf{Index}$
(4) initialize oracles $\Pi_x^{I,J,t}$;
(5) Run adversary $\mathcal{A}$:
(6) - decrypt queries to oracles $\Pi_x^{IJ} \neq \mathcal{O}$
using the secret key of I
or the decryption oracle;
answer the queries as in Figure 3
(7) - answer the $\mathcal{O} : \mathsf{Send}(c_1)$ as follows
(8) parse $\mathsf{Dec}_{sk_B}(c_1)$ as A, n_A
(9) $n_B^0, n_B^1 \xleftarrow{R} \{0,1\}^k$, $n_B^0 \neq n_B^1$
(10) $c_2 \xleftarrow{R} \mathsf{Enc}_{pk}(\mathrm{LR}((A, n_A, n_B^0), (A, n_A, n_B^1), b))$
(11) return c_2 to $\mathcal{A}$
(12) - when $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}(c_3)$
(13) parse $\mathsf{Dec}_{sk}(c_3)$ as n_B
(14) if $n_B = n_B^1$ then $d \leftarrow 1$
(15) if $n_B = n_B^0$ then $d \leftarrow 0$
(16) else $d \xleftarrow{R} \{0,1\}$;
(17) return d

Figure 8. Construction of adversary $\mathcal{B}_4$ against $\mathcal{AE}$ starting from an adversary $\mathcal{A}$ against $\mathsf{NSL}[\mathcal{AE}]$

oracle $\mathcal{O}$. When the first query is made, $\mathcal{B}_4$ decrypts the query using the secret key of B, and obtains A, n_A. Next, it generates two nonces n_B^0, n_B^1 and submits to the lr-oracle the query (m_0, m_1), where $m_i = (B, n_A, n_B^i)$. The ciphertext that is obtained from the lr-oracle, i.e. the encryption of m_b under pk is returned to $\mathcal{A}$ as answer to its query. Since $\mathcal{A}$ makes oracle $\mathcal{O}$ accept, at some latter point $\mathcal{A}$ must make a query to $\mathcal{O}$ that is the encryption of n_B^b; $\mathcal{B}_4$ can precisely determine that selection bit by decrypting this query. The probability that this happens is equal to $\Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]$. Notice that if the nonce is not correctly recovered, the adversary still has 1/2 chances to give the right answer by flipping the bit d.

So, for each $b = 0, 1$ we have that:

$$\begin{aligned}
&\Pr\left[\,\mathbf{Exp}^{\text{ind-cca_}b}_{\mathcal{AE},\mathcal{B}_4}(k) = b\,\right] \\
&= \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right] + \frac{1}{2} \cdot (1 - \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]) \\
&= \frac{1}{2} \cdot (1 + \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right]) \qquad (10)
\end{aligned}$$

We can now compute the advantage of $\mathcal{B}_4$:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_4}(k) &= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_4}(k) = 1\,\right] - \\
&\quad \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},\mathcal{B}_4}(k) = 1\,\right] \\
&= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},\mathcal{B}_4}(k) = 1\,\right] + \\
&\quad \Pr\left[\,\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},\mathcal{B}_4}(k) = 0\,\right] - 1 \\
&= \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E}_{\mathsf{ask}}}\,\right] \qquad (11)
\end{aligned}$$

The last case we consider is that event $\mathsf{E}_{\mathsf{sask}}$ occurs. Let $\mathcal{O}' = \Pi_i^{BA,t}$ the oracle to which $\mathcal{A}$ queries c_2, and denote by $\overline{c_1}$ and $\overline{c_3}$ the first and the second message that O' outputs, i.e. the answers to the queries $\mathcal{O}' : \mathsf{Send}()$ and $\mathcal{O}' : \mathsf{Send}(c_2)$ respectively. We will show that $c_1 = \overline{c_1}$ and $c_3 = \overline{c_3}$, i.e. $\mathcal{O}$ and $\mathcal{O}'$ have matching conversations. Some more notation is in order: we denote by E_{fm} the event that the first message sent by $\mathcal{O}'$ differs from the first message received by $\mathcal{O}$, i.e. $c_1 \neq \overline{c_1}$. We denote by E_{sm} the event that the second message output by $\mathcal{O}'$ is different from the second message received by $\mathcal{O}$, i.e. $c_3 \neq \overline{c_3}$. We obtain:

$$\Pr\left[\,\mathsf{E}_{\mathsf{sask}} \wedge \mathsf{E}_{\mathcal{O}}\,\right] \leq \Pr\left[\,\mathsf{E}_{\mathsf{sask}} \wedge (\mathsf{E}_{\mathsf{fm}} \vee \mathsf{E}_{\mathsf{sm}})\,\right] \qquad (12)$$

The rest of the proof consists of the construction of three adversaries $\mathcal{B}_5, \mathcal{B}_6, \mathcal{B}_7$ having their probability of success related with the probability of occurrence of events $\mathsf{E}_{\mathsf{sask}} \wedge \mathsf{E}_{\mathsf{fm}}$ and $\mathsf{E}_{\mathsf{sask}} \wedge \mathsf{E}_{\mathsf{sm}}$. Due to space constraints, we will detail adversary $\mathcal{B}_5$ and only state the result concerning $\mathcal{B}_6$ and $\mathcal{B}_7$. We will provide a detailed description of these two adversaries in the full version of this paper.

Adversary $\mathcal{B}_5$ is based on the intuition that the adversary manages to create ciphertexts c_1 and $\overline{c_1}$ that encrypt (under the public key of B) the same message, (A, n_A).

The CCA adversary $\mathcal{B}_5$ in Figure 9 exploits this property of $\mathcal{A}$. It starts out by setting up the same environment as before, i.e. it obtains public/secret keys for all parties, except party B. The encryption key of party B is set to be pk (the key obtained by $\mathcal{B}_5$ from the CCA experiment in which it runs). At this point, $\mathcal{B}_5$ chooses a random initiator oracle $\mathcal{O}'$ among the oracles of type $\Pi_i^{BA,t}$. With high probability (1 in N_s) the oracle selected is the oracle to which adversary $\mathcal{A}$ will query message c_2. Next, adversary $\mathcal{B}_5$ runs adversary $\mathcal{A}$ answering its queries as already described for our previous adversaries. When $\mathcal{B}_5$ intercepts the first query of $\mathcal{A}$ to oracle $\mathcal{O} = \Pi_i^{AB,s}$, adversary $\mathcal{B}_5$ selects two random nonces n_A^0, n_A^1 and queries to the lr-oracle the pair (m_0, m_1) where $m_i = (A, n_A^i)$. It obtains in return c_1, the encryption of (A, n_A^b) under the public key of B, and passes this ciphertext as answer to $\mathcal{A}$. It then continues the simulation, until $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}(\overline{c_1})$, at which point $\mathcal{B}_5$ submits $\overline{c_1}$ to the decryption oracle and obtains (A, n_A). By simply comparing n_A with n_A^0, n_A^1, $\mathcal{B}_5$ determines the selection bit of the lr-oracle. We emphasize that by the assumption that $c_1 \neq \overline{c_1}$, the adversary $\mathcal{B}_5$ can safely query $\overline{c_1}$ to the decryption oracle (since it was not obtained as result of a query to the lr-oracle). If n_A is different than both nonces, the answer of $\mathcal{B}_5$ is a random bit.

Adversary $\mathcal{B}_5^{\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,b)),\mathsf{Dec}_{sk}(\cdot)}$

(1) For $I \in \mathsf{IDs} - \{B\}$
(2) $(pk_I, sk_I) \xleftarrow{R} \mathsf{Kg}(k)$
(3) $pk_B \leftarrow pk$
(4) For $(I, J, s, x) \in \mathsf{Index}$
(5) Initialize oracles $\Pi_x^{IJ,s}$
(6) $s \xleftarrow{R} \mathsf{SNo}$
(7) Run adversary $\mathcal{A}$:
(8) - decrypt queries to oracles $\Pi^{IJ,t} \neq \mathcal{O}$
using the secret key of I
or the decryption oracle;
answer the queries as in Figure 3
(9) - when $\mathcal{A}$ makes the query $\Pi_i^{BA,s} : \mathsf{Send}()$
(10) $n_A^0, n_A^1 \xleftarrow{R} \{0,1\}^k$
(11) $c_1 \leftarrow \mathsf{Enc}_{pk}(\mathrm{LR}((A, n_A^0), (A, n_A^1), b))$
(12) return c_1 to $\mathcal{A}$
(13) - when $\mathcal{A}$ makes the query $\mathcal{O} : \mathsf{Send}(c)$
(14) parse $\mathsf{Dec}_{sk_B}(c)$ as A, n_A ;
(15) if $n_A = n_A^0$ then $d \leftarrow 0$
(16) if $n_A = n_A^1$ then $d \leftarrow 1$
(17) else $d \xleftarrow{R} \{0,1\}$

Figure 9. Construction of adversary $\mathcal{B}_5$ against CCA security of $\mathcal{AE}$, from an adversary $\mathcal{A}$ against $\mathsf{NSL}[\mathcal{AE}]$.

If event $\mathsf{E_{sask}}$ occurs, there exists a unique oracle $\mathcal{O}' = \Pi_i^{BA,t}$ which is queried with ciphertext c_2; this oracle is selected in line (6) of the algorithm with probability $\frac{1}{N_s}$. Since the message $\overline{c_1}$ output by $\mathcal{O}'$ as result of $\mathsf{Send}()$, and the message c_1 queried by $\mathcal{A}$ to $\mathcal{O}$ encrypt the same plaintext, i.e. A, n_A^b, adversary $\mathcal{B}_5$ correctly n_A^b provided that his guess with regard to the identity of $\mathcal{O}'$ is correct, and events $\mathsf{E_{sask}}$ and $\mathsf{E_{fm}}$ occur. For each bit $b = 0, 1$:

$$\Pr\left[\mathbf{Exp}^{\text{ind-cca}\_b}_{\mathcal{AE},\mathcal{B}_5}(k) = b\right] \geq$$

$$\frac{1}{\mathsf{N}_s} \cdot \Pr[\,\mathsf{E_{sask}} \wedge \mathsf{E_{fm}}\,] + \frac{1}{2}\left(1 - \frac{1}{N_s} \cdot \Pr[\,\mathsf{E_{sask}} \wedge \mathsf{E_{fm}}\,]\right) =$$

$$\frac{1}{2} + \frac{1}{2N_s}\Pr[\,\mathsf{E_{sask}} \wedge \mathsf{E_{fm}}\,]$$

computing the advantage of $\mathcal{B}_5$ as before we obtain

$$N_s \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_5}(k) \geq \Pr[\,\mathsf{E_{sask}} \wedge \mathsf{E_{fm}}\,] \qquad (13)$$

In the full version of the paper we will show how to construct two adversaries $\mathcal{B}_6, \mathcal{B}_7$ such that:

$$\Pr[\,\mathsf{E_{sask}} \wedge \mathsf{E_{sm}}\,] \leq$$

$$N_s \cdot \left(\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_6}(k) + \frac{1}{2}\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_7}(k)\right) \qquad (14)$$

Putting together Equations (12), (13) and (14), by standard probability theory we obtain

$$\begin{aligned}
\Pr[\,\mathsf{E}_{\mathcal{O}}\,] &= \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E_{ask}}}\,\right] + \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E_{sask}}\,] + \\
&\quad \Pr[\,\mathsf{E}_{\mathcal{O}} \wedge \mathsf{E_{mask}}\,] \\
&\leq \Pr\left[\,\mathsf{E}_{\mathcal{O}} \wedge \overline{\mathsf{E_{ask}}}\,\right] + \Pr[\,\mathsf{E_{fm}} \wedge \mathsf{E_{sask}}\,] + \\
&\quad \Pr[\,\mathsf{E_{sm}} \wedge \mathsf{E_{sask}}\,] + \Pr[\,\mathsf{E_{mask}}\,] \\
&\leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_4}(k) + N_s \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_5}(k) + \\
&\quad N_s \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_6}(k) + \frac{N_s}{2} \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},\mathcal{B}_7}(k)
\end{aligned}$$

Since the function on the left side is non-negligible, so is at least one of the terms on the right side, which in turn implies that $\mathcal{AE}$ is insecure. This completes our proof. ∎

6 Discussion

This paper provides a computational proof that the Needham-Schröeder-Lowe protocol is a secure mutual authentication protocol, if the encryption scheme used in the implementation is IND-CCA secure. The framework used in our proof was introduced in [5], and is considered typical for the computational approach. Mutual authentication is captured in this model using the technical notion of "matching conversations". The formulation is identical in spirit to the formulation of mutual authentication used in the strand space model [9]. Indeed, in both approaches, protocol executions can be viewed as graphs (labeled with bit-strings, or terms from an appropriate term algebra) satisfying a well-formedness condition, and in both cases the security requirement is a requirement about these graphs. Moreover, although the techniques used to prove security with respect to these definitions are different (reductions and logical reasoning about relations, respectively), the structure of the proofs is the same in both cases.

A challenging problem is to relate real and formal executions of protocols in order to obtain a general result stating that a proof that mutual authentication is satisfied in the formal model implies that it is also satisfied in the concrete world. A statement along these lines was proved for the case of authenticated message delivery in [12].

A natural question is whether IND-CCA is the weakest security notion for encryption schemes that ensures security of the NSL protocol. An answer to this question would have important implications for the relation between formal and computational treatments of security. We first point out a somewhat disturbing fact about the definition of IND-CCA security. Consider an encryption scheme that is IND-CCA secure (see Section 2.) Modify the encryption function by adding an extra bit to the normally obtained ciphertext, set this bit randomly to 0 or 1. The new decryption function ignores the bit, and obtains the plaintext by applying the

decryption algorithm of the original scheme. Surprisingly, although the new encryption scheme does not reveal more information about the plaintext than the original one, the new scheme is no longer IND-CCA secure: if cb is the challenge ciphertext (where the last bit of the ciphertext b is explicitly shown), one can immediately recover the underlying the plaintext by submitting to the decryption oracle the ciphertext $c\bar{b}$. This example shows that IND-CCA may be unnecessarily strong.

To cope with this definitional inadequacy, a new security notion, termed *benign malleability* or g-CCA2 security, has been recently introduced [20, 2]. The associated security definition considers an adversarial model slightly weaker than the one for IND-CCA: the adversary has access to the decryption oracle, but its queries are restricted to ciphertexts for which the underlying plaintext is different from the one underlying the challenge ciphertext. We claim that this new security notion is better suited for implementing formal encryption. Indeed, denying the adversary the ability to create a ciphertext c' from a ciphertext c having the same plaintext seems to be irrelevant for all practical security purposes.

Let us now return to the NSL protocol. We claim that Definition 4.3 rules out as insecure protocols that are intuitively secure. To see this, consider NSL implemented with a benignly malleable encryption scheme. The resulting protocol is insecure (according to the given definition) since an adversary that relays messages between oracles, changing the ciphertexts without modifying the underlying plaintext has advantage 1. This indicates that the security definition for mutual authentication might be too strong. We thus consider a slightly weaker security definition in which oracles are required to accept, if and only if they have matching conversations at the *plaintext* level (rather than at *ciphertext* level). The proof that NSL is secure with respect to this new definition when encryption is gCCA2 secure seems to be greatly simplified. In particular, the technical requirement that ciphertexts obtained from the lr-oracles are never queried to the decryption oracle is trivially satisfied. The precise details are being worked out.

Acknowledgements. The presentation of our results was substantially improved following discussions with Alexandra Boldyreva and Adriana Palacio.

References

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

[2] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Theory and Application of Cryptographic Techniques*, pages 83–107, 2002.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. of EUROCRYPT'00*, LNCS, 2000.

[4] M. Bellare and P. Rogaway. Introduction to modern cryptography-Lecutre notes. Available at: http://www.cs.ucsd.edu/users/mihir/cse207/classnotes.html.

[5] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, 1994.

[6] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proc. of 27th Anual Symposium on the Theory of Computing*. ACM, 1995.

[7] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[8] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[9] F. J. T. Fabrega, J. Hertzog, and J. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[10] O. Goldreich. *Foundations of Cryptography*. 2001.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *J. of Computer and System Sciences*, 28:270–299, April 1984.

[12] J. Guttman, F. J. T. Fabrega, and L. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 186–195, 2001.

[13] G. Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information processing letters*, 56(3):131–136, November 1995.

[14] G. Lowe. Breaking and fixing the Needham-Schröeder algorithm. In *Proc. of TACAS'96*, pages 147–166. Springer-Verlag, 1996.

[15] F. S. E. Ltd. Failures divergence refinement - user manual and tutorial, 1993. Version 1.3.

[16] C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[17] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of ACM*, December 1978.

[18] B. Pfitzmann and M. Waidner. A model for asynchronous reactive sysmtes and its applications to secure message transmission. In *Proceedings of the 22nd IEEE Symposium on Security and Privacy*, pages 184–200, 2001.

[19] C. Rackoff and D. Simon. Noninteractive zero-knowledge proofs of knowledfe and chosen ciphertext attack. In *Advances in Cryptology- CRYPTO'91*, pages 433–444, 1991.

[20] V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). Manuscript, Dec. 20, 2001.

[21] D. Song. An automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, June 1999.