Computationally Complete Symbolic Attacker in Action

Gergei Bana[1], Pedro Adão[2], and Hideki Sakurada[3]

[1] MSR-INRIA Joint Centre, Orsay, France bana@math.upenn.edu
[2] SQIG-IT and IST-TULisbon, Portugal, pedro.adao@ist.utl.pt
[3] NTT Communication Science Laboratories, Atsugi, Kanagawa, Japan, sakurada.hideki@lab.ntt.co.jp

Abstract. We show that the recent technique of computationally complete symbolic attackers proposed by Bana and Comon-Lundh [7] for computationally sound verification of security protocols is powerful enough to verify actual protocols. In their work, Bana and Comon-Lundh presented only the general framework, but they did not introduce sufficiently many axioms to actually prove protocols. We present a set of axioms—some generic axioms that are computationally sound for all PPT algorithms, two specific axioms that are sound for CCA2 secure encryptions, and a further minimal parsing assumption for pairing without which there is an attack—and illustrate the power of this technique by giving the first computationally sound verification (secrecy and authentication) via symbolic attackers of the NSL Protocol that does not need any further restrictive assumptions about the computational implementation. In other words, all implementations for which the axioms are sound – namely, implementations using CCA2 encryption, and satisfying the parsing requirement for pairing – exclude the possibility of successful computational attacks. Furthermore, the axioms are entirely modular, not particular to the NSL protocol.

1 Introduction

Computational soundness has been a topic of utmost importance in the past decade. It started with Abadi and Rogaway [1] and followed by many others, for passive adversaries [16, 2] as well as for active. The aim is always that symbolic proofs imply computational security. Works concerning active adversaries can be divided into two groups. Works in one [3, 14, 4, 12] define symbolic adversaries, and soundness theorems state that under certain circumstances, if there is no successful symbolic attack, then there is no successful computational attack. The other group aims to work directly in the computational model [15, 8, 11, 10].

The first group, where symbolic attacker is defined, gives hope that already existing automated tools may be used for computationally sound verification, but these soundness theorems require a large set of assumptions. Typically they assume that no key cycles can ever be created, that bitstrings can be unambiguously parsed into terms, that there is no dynamic corruption, that keys are certified (badly generated keys cannot be used), etc. These assumptions, as well as reasons why they are not realistic are discussed in [13].

Recently, Bana and Comon-Lundh presented in [7] (and in an improved version [6]) a new technique to define symbolic attackers that is more suitable for computational soundness than the usual Dolev-Yao adversary (or variations of it). They called this new symbolic adversary *computationally complete symbolic adversary*, as it is capable of doing everything that a computational adversary is capable of. The basic idea of their technique is the following. Instead of listing every kind of move a symbolic adversary is allowed to do, a few rules (axioms) should be listed that the symbolic adversary is not allowed to violate. In other words, the symbolic adversary is allowed to do everything that is consistent with these axioms. The axioms that are introduced must be computationally sound with respect to a computational interpretation that they defined. Their main result is that once it is shown that no successful symbolic adversary can exist complying some set of axioms, then for any computational implementation satisfying that set of axioms, successful computational attacks are impossible as long as the number of sessions is bounded in the security parameter.

In their original work however, they did not show that their technique could actually be used for practical protocol verification as they only presented the general framework and a few computationally sound axioms as a proof of concept. To actually prove protocols, more axioms have to be introduced in order to *weaken* the symbolic adversary sufficiently close to the computational adversary (with unrealistically strong symbolic attackers, no protocol can be verified). They left it for future work to show that this program could be carried out in practice.

In this paper, we show the technique of Bana and Comon-Lundh can indeed be used for protocol verification. Namely, we introduce some modular, computationally sound axioms, and then illustrate that the technique with the axioms we introduced can be used to verify secrecy and authentication of an actual protocol, namely, the Needham-Schroeder-Lowe protocol. More precisely, we show that there is no symbolic adversary for which the violation of either secrecy or authentication (or both) of the NSL protocol is consistent with the set of general axioms we give together with an additional minimal parsing property that the computational implementation of *pairing* must satisfy (otherwise there is an attack). Applying the main theorem of [7], this means that there is no computational adversary of the NSL protocol (in an implementation satisfying the axioms) that can violate secrecy or authentication with non-negligible probability.

The set of axioms we give is divided into four groups. One has a number of general axioms sound for any computational implementation. Then, there is a group consisting of the equations required for function symbols, such as the decryption of a cipher gives the plaintext back. Then, there is a group of two axioms sound for CCA2 encryptions, one expressing the secrecy of a CCA2 encrypted item, and one expressing the non-malleability property of CCA2 encryptions. Furthermore, to prove security of the NSL protocol, one more property is needed expressing a certain parsing unambiguity, which needs to be assumed as otherwise an attack exists.

The axioms are not particular to NSL protocols. They are modular. Introducing further primitives will not destroy the soundness of these axioms, they do not have to be proved again.

The technique of [7] allows to avoid *all* restrictions mentioned before on the computational world. *Once a protocol is proven secure in our symbolic model with respect to a set of axioms, then all properties that the computational implementation has to satisfy for computational security are included in the axioms.* Any number of bad keys are allowed to be generated by the adversary; any number of corrupted, uncorrupted, or dynamically corrupted parties can be present. As for parsing of bit strings into terms, previous soundness results relied on unambiguous parsing. Within this framework, we do not need such an assumption, because *symbolic agents do not do any pattern matching*. If unambiguous parsing is needed for the security of a protocol, then it is necessary to list it as a property that secure implementations need to satisfy. The only needed assumption for proving NSL is that an honestly generated nonce N cannot be non-negligibly parsed into a pair, such that the second part of the pair is some (dishonest) agent name, *i.e.*, looks like $\langle n, Q\rangle$ for some n. This is a necessary assumption, as the failure of it results in an attack, presented in [9]. This can easily be achieved in an implementation by, for example, checking the length of bit strings that should correspond to nonces. Other than this, no parsing hypothesis is assumed. For example, honestly generated nonces may collide with other kinds of pairs, encryptions could *a priori* collide with other kinds of expressions, etc. That is, *tagging of pairs, encryption etc is not necessary for the security of the NSL protocol*.

In fact, the security proof is long exactly because of parsing ambiguities. Since any term may *a priori* coincide with any other, that is, any term that was created by an honest agent or the adversary may *a priori* be wrongly parsed by another agent (interpreting it as the message of a wrong session or a wrong message of a given session), the fact that they do not, has to be derived from the nature of the protocol. Had we assumed tagging and completely unambiguous parsing (which, in fact, has always been assumed by earlier NSL proofs, even in Cryptoverif), the proof would be quite short.

We would like to emphasize that our aim here is to demonstrate that the technique works, and not to provide the most general possible verification for the NSL protocol. Further generalizations are possible at the cost of much longer proofs. For example, in the current proof, we assume that each honest agent only executes either the initiator or the responder role as this makes the proof much shorter and clearer. We have however been able to complete the proof for the case when they are allowed to run both sessions, even against themselves at the cost of an additional parsing axiom. A further assumption is that triples are created from pairs. It is possible to do the proofs without this assumption, and have a separate function symbol for triples (and introduce further necessary requirements to avoid attacks), but again, it would make the proofs far longer.

The contributions of this paper include (i) the set of general axioms that are computationally sound for any PPT implementation, (ii) the non-malleability axiom that is computationally sound for CCA2 security, (iii) the additional parsing axiom needed to avoid an attack to the NSL, and (iv) the security proof itself. Again, the axioms are all modular, independent of the protocol, and they do not have to be proved again if further primitives are included.

This paper is organized as follows: we start by recalling the framework of [7] (Section 2). In Section 3, we show how the NSL protocol and its execution can be formulated in the proposed framework. In Section 4, we present the first contribution of this paper introducing the set of computationally sound axioms needed to show both secrecy and authentication of the NSL protocol. In Section 5 we show soundness of the two non-trivial axioms, namely, the CCA2 axioms. In Section 6, we show a few simple examples of how inconsistency of certain formulas with the axioms can be proven. In Section 7, we prove that no symbolic adversary compliant with the presented axioms can successfully violate secrecy or authentication of the NSL protocol. In Section 8, we summarize our results and present directions for future work.

2 Symbolic Execution

The framework used in this paper was introduced by Bana and Comon-Lundh in [7]. We resent a brief summary of that (with minor adjustments) and refer the reader to that paper for further details.

2.1 Terms and Frames

Terms are built out of a set of function symbols $\mathcal{F}$ that contains an unbounded set of names $\mathcal{N}$ and an unbounded set of handles $\mathcal{H}$. Names and handles are zero-arity function symbols. We will use names to denote items honestly generated by agents, while handles will denote inputs of the adversary. Let $\mathcal{X}$ be an unbounded set of variables. A ground term is a term without variables. *Frames* are sequences of terms together with name binders: a frame ϕ can be written $(\nu\overline{n}).p_1 \mapsto t_1, \ldots, p_n \mapsto t_n$ where $p_1, \ldots p_n$ are place holders that do not occur in $t_1, \ldots, t_n$ and $\overline{n}$ is a sequence of names. $\mathsf{fn}(\phi)$, the *free names* of ϕ are names occurring in some t_i and not in $\overline{n}$. The *variables* of ϕ are the variables of $t_1, \ldots, t_n$.

2.2 Formulas

Let $\mathcal{P}$ be a set of predicate symbols over tems. $\mathcal{P}$ is assumed to contain the binary predicate $=$ (which is interpreted as a congruence) and is used as $t_1 = t_2$, and a family of $n+1$ predicates $\vdash_n$, (which is intended to model the adversary's capability to derive something) and is used as $t_1, ..., t_n \vdash t$. (we drop the index n for readability).

As for the symbolic interpretation of such predicates, we *allow any* that does not contradict our axioms, which we will introduce later.

Let $\mathcal{M}$ be any first-order structure that interprets the function and predicate symbols of the logic. We only require that it interprets terms and the predicates such that $=$ is interpreted as the equality in the underlying domain $D_{\mathcal{M}}$ (clearly, $D_{\mathcal{M}}$ includes a relation $\vdash_{\mathcal{M}}$ interpreting the deducibility predicate $\vdash$ too). Given an assigment σ of elements in $D_{\mathcal{M}}$ to the free variables of term t, we write $[\![t]\!]^{\sigma}_{\mathcal{M}}$ for the interpretation of $t\sigma$ in $\mathcal{M}$ ($[\![\_]\!]^{\sigma}_{\mathcal{M}}$ is the unique extension of σ into a homomorphism of $\mathcal{F}$-algebras). For any first order structure $\mathcal{M}$ over the functions $\mathcal{F}$ and predicates $\mathcal{P}$, the satisfaction relation $\mathcal{M}, \sigma \models \theta$, where σ is an assignment of the free variables of θ in the domain of $\mathcal{M}$, is defined as usual in first-order logic.

2.3 Execution of a Protocol

Definition 1. *A* symbolic state *of the network consists of:*

- *a control state $q \in Q$ together with a sequence of names (that have been generated so far) $n_1, \ldots, n_k$*
- *a sequence constants called* handles $h_1, \ldots, h_n$ *(recording the attacker's inputs)*
- *a ground frame ϕ (the agents outputs)*
- *a set of formulas Θ (the conditions that have to be satisfied in order to reach the state).*

A *symbolic transition sequence* of a protocol Π is a sequence

$$(q_0(\overline{n_0}), \emptyset, \phi_0, \emptyset) \to \ldots \to (q_m(\overline{n_m}), \langle h_1, \ldots, h_m \rangle, \phi_m, \Theta_m)$$

if, for every $m-1 \geq i \geq 0$, there is a transition rule

$$(q_i(\overline{\alpha_i}), q_{i+1}(\overline{\alpha_{i+1}}), \langle x_1, \ldots, x_i \rangle, x, \psi, s)$$

such that $\overline{n} = \overline{\alpha_{i+1}} \setminus \overline{\alpha_i}$, $\phi_{i+1} = (\nu\overline{n}).(\phi_i \cdot p \mapsto s\rho_i\sigma_{i+1})$, $\overline{n_{i+1}} = \overline{n_i} \uplus \overline{n}$, $\Theta_{i+1} = \Theta_i \cup \{\phi_i \vdash h_{i+1}, \psi\rho_i\sigma_{i+1}\}$ where $\sigma_i = \{x_1 \mapsto h_1, \ldots, x_i \mapsto h_i\}$ and ρ_i is a renaming of the sequence $\overline{\alpha_i}$ into the sequence $\overline{n_i}$. We assume a renaming that ensures the freshness of the names $\overline{n}$: $\overline{n} \cap \overline{n_i} = \emptyset$.

Definition 2. *Given an interpretation $\mathcal{M}$, a transition sequence of Π*

$$(q_0(\overline{n_0}), \emptyset, \phi_0, \emptyset) \to \ldots \to (q_m(\overline{n_m}), \langle h_1, \ldots, h_m \rangle, \phi_m, \Theta_m)$$

is valid w.r.t. $\mathcal{M}$ *if, for every $m-1 \geq i \geq 0$, $\mathcal{M} \models \Theta_{i+1}$.*

Initialization. For technical purposes, we always take $\phi_0 = \nu\overline{n}()$, where $\overline{n}$ contains the honest random items to be generated, and that is followed by an empty list of terms. ϕ_1 will contain the output of the initialization, that is, the names and the public keys. We will also assume for technical purposes that all honestly generated items (nonces, random inputs of encryptions etc.) are generated upfront.

2.4 Satisfaction of Predicates, Constraints and FOL Formulas in Executions

$\mathcal{M}$ modeled, among others, the predicate $t_1, ..., t_n \vdash t$. In *executions* however, instead of this predicate, we consider a predicate that we write as $\hat{\phi}, t_1, ..., t_n \vdash t$. This is also an $n+1$-arity predicate. $\hat{\phi}$ is just a symbol, not an argument, and it represents the frame containing the messages that protocol agents sent out, that is, the information available from the protocol to the adversary. Computational semantics of the predicates $x = y$ and $\hat{\phi}, x_1, ..., x_m \vdash x$ were defined in [7]. Here we just briefly mention that $=$ refers to equality up to negligible probability, and $\vdash$ means that the adversary is able to compute (with a PT algorithm) the right side from the left. We also use another predicate, $W(x)$, which just tells if x is the name of an agent. We also use 4 different *constraints*: $\mathsf{Handle}(h)$, and $\mathsf{RandGen}(x)$, and $x \sqsubseteq \hat{\phi}$, and $x \sqsubseteq \boldsymbol{x}$. $\mathsf{Handle}(h)$ means h is a handle, $\mathsf{RandGen}(x)$ means that x was honestly, randomly generated (i.e. appearing under ν in the frame); $x \sqsubseteq \hat{\phi}$ means that x was part of a message sent out by an agent (i.e. listed in the frame ϕ), $x \sqsubseteq \boldsymbol{x}$ means x is part of $\boldsymbol{x}$. Let us introduce the following abbreviations:

- $x \sqsubseteq \hat{\phi}, \boldsymbol{x} \;\equiv\; x \sqsubseteq \hat{\phi} \vee x \sqsubseteq \boldsymbol{x}$
- $\mathsf{fresh}(x; \hat{\phi}, \boldsymbol{x}) \;\equiv\; \mathsf{RandGen}(x) \wedge x \not\sqsubseteq \hat{\phi} \wedge x \not\sqsubseteq \boldsymbol{x}$
- $\boldsymbol{x} \preccurlyeq \hat{\phi} \;\equiv\; h \sqsubseteq \boldsymbol{x} \wedge \mathsf{Handle}(h) \rightarrow \hat{\phi} \vdash h$

Given a first-order model $\mathcal{M}$ as before, satisfaction of predicates and constraints in a *symbolic* execution is defined as:

- Interpretation of predicates by $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k)$, where σ is a substitution as above, $t_1, ..., t_m$ are closed terms, and $n_1, ..., n_k$ are names: (note the interpretation depends on $\mathcal{M}$) is defined as follows
  - $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k) \models t = t'$ if $\mathcal{M}, \sigma \models t = t'$
  - $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, \overline{n} \models \hat{\phi}, s_1, \ldots, s_n \vdash t$ if $\mathcal{M}, \sigma \models s_1, \ldots, s_n, t_1, \ldots, t_m \vdash t$.
  - $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, \overline{n} \models W(x)$ if $\mathcal{M}, \sigma \models W(x)$
- Interpretations of constraints by $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k)$, where σ is a substitution as above, $t_1, ..., t_m$ are closed terms, and $n_1, ..., n_k$ are names: (do not depend on the model $\mathcal{M}$):
  - $\mathsf{Handle}(h)$ for h closed term:
    $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k) \models \mathsf{Handle}(h)$ if $h \in \mathcal{H}$.
  - $\mathsf{RandGen}(s)$ for s closed term:
    $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k) \models \mathsf{RandGen}(s)$ if $s \in \mathcal{N}$ and $\mathcal{M}, \sigma \models s = n_1 \vee \ldots \vee s = n_k$.
  - $t \sqsubseteq \hat{\phi}$, where t is closed term:
    $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, \overline{n} \models t \sqsubseteq \hat{\phi}$ if t is a subterm of some t_i
  - $t \sqsubseteq s_1, ..., s_n$, where $s_1, ..., s_n$ and t are closed terms:
    $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, \overline{n} \models t \sqsubseteq s_1, ..., s_n$ if t is a subterm of some s_i
- Interpretation of any FOL formula in which there are no free variables under constraints by $\mathcal{M}, \sigma, \langle t_1, \ldots, t_m\rangle, (n_1, \ldots, n_k)$ where σ is a substitution as above, is defined recursively as:

  - Interpretations of $\theta_1 \wedge \theta_2$, $\theta_1 \vee \theta_2$, and $\neg\theta$ are defined as usual in FOL
  - If x is not under a constraint in θ, interpretations of $\forall x\theta$ and $\exists x\theta$ are defined as usual in FOL.
  - If x occurs under a constraint in θ, then
    * $\mathcal{M}, \sigma, \langle t_1, \ldots, t_n\rangle, (n_1, \ldots, n_k) \models \forall x\theta$ iff for every ground term t, $\mathcal{M}, \sigma, \langle t_1, \ldots, t_n\rangle, (n_1, \ldots, n_k) \models \theta\{x \mapsto t\}$
    * $\mathcal{M}, \sigma, \langle t_1, \ldots, t_n\rangle, (n_1, \ldots, n_k) \models \exists x\theta$ iff there is a ground term t, $\mathcal{M}, \sigma, \langle t_1, \ldots, t_n\rangle, (n_1, \ldots, n_k) \models \theta\{x \mapsto t\}$
- Satisfaction at step m: $\mathcal{M}, (q, \langle h_1, \ldots, h_m\rangle, \overline{n}, \phi_m, \Theta) \models \theta$ iff $\mathcal{M}, \phi_m, \overline{n} \models \theta$.

3 The NSL Protocol and Its Symbolic Execution

We now formulate the NSL protocol and its execution in the above framework. The steps of the protocol, as usual are

1. $A \to B\ :\ \{N_1, A\}_{eK_B}$ 2. $B \to A\ :\ \{N_1, N_2, B\}_{eK_A}$ 3. $A \to B\ :\ \{N_2\}_{eK_A}$

We use a randomized public-key encryption symbol: $\{m\}^r_{eK_Q}$ is intended to represent the encryption of the plaintext m with the public-key of the principal Q, with a random seed r. So, consider the a set of constructors $\mathcal{F}_c = \{\{\_\}^{\_}_{\_}, \langle\_, \_\rangle, e_{\_}, d_{\_}, K_{\_}\}$, and a set of destructors $\mathcal{F}_d = \{dec(\_, \_), \pi_1(\_), \pi_2(\_)\}$, with the following equations:

- Equations for encryption/decryption:
  - Decryption of an encryption results the plaintext: $dec(\{x\}^R_{eK}, dK) = x$
- Equations for pairing/projections:
  - First projection: $\pi_1(\langle x, y\rangle) = x$
  - Second projection: $\pi_2(\langle x, y\rangle) = y$

We will use pairs to construct triples: $\langle x, y, z\rangle \equiv \langle x, \langle y, z\rangle\rangle$.

We can define the action of principals as follows: the initiator, communicating with intended party Q, does the following sequence of steps in session i which we will informally denote by $Init^A_{NSL}[A, i, Q, N_1, h_1, h_3, R_1, R_3]$:

1. Receives some handle h_1 from the adversary that triggers the start of the session with intended party Q;
2. A generates nonce N_1;
3. A sends $\{N_1, A\}^{R_1}_{eK_Q}$;
4. A receives h_2, and checks:
   (a) $\pi_1(dec(h_3, dK_A)) = N_1$;
   (b) $\pi_2(\pi_2(dec(h_3, dK_A))) = Q$;
5. A sends $\{\pi_1(\pi_2(dec(h_3, dK_A)))\}^{R_3}_{eK_Q}$;
6. A sends $c_i\big(A, Q, N_1, \pi_1(\pi_2(dec(h_3, dK_A)))\big)$.

We used notation $q^A_i(\overline{n})$ to denote the reached states. For verification purposes, let c_i be a special function symbol, that takes as arguments A, B, N_1, N_2, respectively who commits for whom and the corresponding nonces. $c_i(A, B, N_1, N_2)$ is sent along with $\{N_1, N_2, B\}_A$. For the responder, there is a similar commitment: at the end of the protocol, B emits (as a last message) $c_r(A, B, N_1, N_2)$.

The responder does the following sequence of steps in session i' which we will informally denote by $Resp^B_{NSL}[B, i', N_2, h_2, h_4, R_2]$:

1. B receives some h_2 from the adversary and checks:
   – $W(\pi_2\,(dec(h_2, dK_B)))$ (Checks that it is a name of someone);
2. B generates nonce N_2;
3. B sends $\{\pi_1\,(dec(h_2, dK_B))\,, N_2, B\}^{R_2}_{eK_{\pi_2(dec(h_2,dK_B))}}$;
4. B receives h_4, and checks if $dec(h_4, dK_B) = N_2$;
5. B sends $c_r(\pi_2\,(dec(h_2, dK_B))\,, B, \pi_1\,(dec(h_2, dK_B))\,, N_2)$.

3.1 Example Executions of the NSL Protocol

We now show an example of how an initial segment of the NSL execution can look like.

Example 1. We show the beginning of a possible branch in the symbolic execution of NSL.

$$\begin{tikzcd} (q_0, \emptyset, \phi_0, \emptyset) \arrow[r] & (q_1, H_1, \phi_1, \Theta_1) \arrow[r] & (q_2, H_2, \phi_2, \Theta_2) \arrow[r] & (q_3, H_3, \phi_3, \Theta_3) \arrow[r] & (q_4, H_4, \phi_4, \Theta_4) \end{tikzcd}$$

where $\overline{n} = N_1, N_2, R_1, R_2, R_3$, and, with q^A_j, q^B_j counting the states of A and B, $q_0 = (q^A_0, q^B_0)(\overline{n})$, $q_1 = (q^A_1, q^B_0)(\overline{n})$, $q_2 = (q^A_1, q^B_1)(\overline{n})$, $q_3 = (q^A_2, q^B_1)(\overline{n})$ and $q_4 = (q^A_2, q^B_2)(\overline{n})$. In other words, we interleave the actions of A and B, as in an expected execution and assume that the two processes were first activated (if not, we could introduce two transitions activating the processes).

– $\phi_0 = \nu_{K_A K_B AB}()$, $\quad \Theta_0 = \emptyset$
– $H_1 = \emptyset$, $\quad \phi_1 = \nu_{K_A K_B AB}(p_1 \mapsto (A, B, eK_A, eK_B))$, $\quad \Theta_1 = \emptyset$
– $H_2 = \langle h_1 \rangle$, $\quad \phi_2$ extends ϕ_1 with $p_1 \mapsto \{\langle N_1, A\rangle\}^{R_1}_{eK_B}$, $\quad \Theta_2 = \{\phi_1 \vdash h_1\}$
– $H_3 = \langle h_1, h_2 \rangle$,
  ϕ_3 extends ϕ_2 with $p_2 \mapsto \{\langle \pi_1\,(dec(h_2, dK_B))\,, \langle N_2, B\rangle\rangle\}^{R_2}_{eK_{\pi_2(dec(h_2,dK_B))}}$,
  $\Theta_3 = \Theta_2 \cup \{\phi_2 \vdash h_2, W(\pi_2\,(dec(h_2, dK_B)))\}$
– $H_4 = \langle h_1, h_2, h_3 \rangle$,
  ϕ_4 extends ϕ_3 with $p_3 \mapsto \{\pi_1\,(\pi_2\,(dec(h_3, dK_A)))\}^{R_3}_{eK_B}$,
  $\Theta_4 = \Theta_3 \cup \{\phi_3 \vdash h_3, \pi_1\,(dec(h_3, dK_h)) = N_1, \pi_2\,(\pi_2\,(dec(h_3, dK_A))) = B\}$,
– $H_5 = \langle h_1, h_2, h_3, h_4 \rangle$, $\phi_5 = \phi_4$,
  $\Theta_5 = \Theta_4 \cup \{\phi_4 \vdash h_4, dec(h_4, dK_B) = N_2\}$,

Let $\mathcal{M}$ be a model such that $\pi_2\,(dec(h_2, dK_B)) = A$,

$$h_2 =_{\mathcal{M}} \{\langle N_1, A\rangle\}^{R_1}_{eK_B}, \qquad h_3 =_{\mathcal{M}} \{\langle N_1, \langle N_2, B\rangle\rangle\}^{R_2}_{eK_A}, \qquad h_4 =_{\mathcal{M}} \{N\}^{R_3}_{eK_B},$$

and $\vdash_{\mathcal{M}}$ is simply the classical Dolev-Yao deduction relation. Then the execution sequence above is valid w.r.t. $\mathcal{M}$, and this corresponds to the correct execution of the NSL protocol between A and B.

Example 2. There are however other models in which this transition sequence is valid. Let $\mathcal{M}'$ be such that $h_2 =_{\mathcal{M}'} N_1$ and $\phi_1 \vdash_{\mathcal{M}'} h_2$ and $N_1 =_{\mathcal{M}'} \{\langle N_1, A\rangle\}^{R_1}_{eK_B}$, (and h_3, h_4 as above). We get again a valid transition sequence w.r.t. $\mathcal{M}'$. Though, in what follows, we will discard such sequences, thanks to some axioms.

Example 3. Consider again the transitions of the Example 1. Now consider a model $\mathcal{M}$ in which $N_0, \{N_1, N_2, B\}^{R_2}_{eK_A} \vdash_{\mathcal{M}} \{N_1, N_0, B\}^{r}_{eK_A}$ for an honestly generated nonce N_0 that can be chosen by the attacker: the transition sequence of the previous example is also valid w.r.t. this model. This however yields an attack, using a malleability property of the encryption scheme.

Discarding such attacks requires some properties of the encryption scheme (for instance IND-CCA). It can be ruled out by the non-malleability axiom that we will introduce.

From this example, we see that unexpected attacks can be found when some assumption is not explicitly stated as an axiom to limit adversarial capabilities.

4 The Axioms

This section contains the core results of this paper: a set of computationally sound axioms that are sufficient to prove security of actual protocols that use CCA2 secure encryptions. The axioms are not at all special to the NSL protocol and can be used in other protocol proofs too. They are entirely modular, so introducing further primitives will not invalidate their soundness, they do not have to be verified again. As usual, unquantified variables are universally quantified.

– **Equality is a Congruence.** The first axiom says that the equality is a congruence relation:
  - $x = x$, and the substitutability (congruence) property of equal terms holds for predicates (but not necessarily constraints).

  This axiom is computationally sound simply as we limit ourself to consider computational interpretations of predicates that are invariant if we change the arguments on sets with negligible probability. The computational interpretations of $=$, $\vdash$ and W are all such.

– **Axioms for the Derivability Predicate.** The following axioms are trivially computationally sound for what the PPT adversary can compute. The last is sound as we assume that all function symbols are interpreted as PT computable functions.
  - Self derivability: $\hat{\phi}, \boldsymbol{x}, x \vdash x$
  - Increasing capabilities: $\hat{\phi}, \boldsymbol{x} \vdash y \ \longrightarrow \ \hat{\phi}, \boldsymbol{x}, x \vdash y$
  - Commutativity: If $\boldsymbol{x}'$ is a permutation of $\boldsymbol{x}$, then $\hat{\phi}, \boldsymbol{x} \vdash y \ \longrightarrow \ \hat{\phi}, \boldsymbol{x}' \vdash y$
  - Transitivity of derivability: $\hat{\phi}, \boldsymbol{x} \vdash \boldsymbol{y} \ \wedge \ \hat{\phi}, \boldsymbol{x}, \boldsymbol{y} \vdash \boldsymbol{z} \ \longrightarrow \ \hat{\phi}, \boldsymbol{x} \vdash \boldsymbol{z}$
  - Functions are derivable: $\hat{\phi}, \boldsymbol{x} \vdash f(\boldsymbol{x})$

– **Axioms for Randomly Generated Items.** These axioms express relations between $\mathsf{RandGen}()$ and the $\sqsubseteq$ constraints that are not purely symbolic (for example, as $\sqsubseteq$ is a constraint, $x \sqsubseteq \hat{\phi}, x$ holds purely symbolically, so it does not have to be listed as an axiom as opposed to $\hat{\phi}, \boldsymbol{x}, x \vdash x$ which is a predicate). The no telepathy axiom expresses that items that were randomly generated but not yet sent out are not guessable. It is sound because we assumed that random generation happens in a large enough space such that guessing is only possible with negligible probability. The second axiom is sound because random generation is independent of everything else, so a randomly generated item x given to the adversary cannot help to compute y from which it is independent, before x was sent out (that is, appear in ϕ). Once x appears in the frame (as e.g. $\{y\}_x^R$), giving x to the adversary may help to compute y.
  - No telepathy: $\mathsf{fresh}(x; \hat{\phi}) \ \longrightarrow \ \hat{\phi} \not\vdash x$
  - Fresh items are independent and hence contain no information about other items:

$$\mathsf{fresh}(x; \hat{\phi}, \boldsymbol{x}, y) \ \wedge \ \boldsymbol{x} \preccurlyeq \hat{\phi} \ \wedge \ y \preccurlyeq \hat{\phi} \ \wedge \ \hat{\phi}, \boldsymbol{x}, x \vdash y \ \longrightarrow \ \hat{\phi}, \boldsymbol{x} \vdash y$$

– **Equations for the fixed function symbols.** discussed earlier:
  - $dec(\{x\}_{eK}^R, dK) = x; \quad \pi_1(\langle x, y \rangle) = x; \quad \pi_2(\langle x, y \rangle) = y$

– **Special to IND-CCA Encryption.** Let $x_1, ..., x_n \preccurlyeq \hat{\phi} \ \equiv \ x_1 \preccurlyeq \hat{\phi} \ \wedge \ ... \ \wedge \ x_n \preccurlyeq \hat{\phi}$. We present two axioms here. Both follow if the encryption is CCA2 secure and if random generation is only guessable with negligible probability. The first expresses secrecy, the second non-malleability. None of them implies the other. As they are not trivial, we state them in the form of Theorems.
  - **Secrecy of CCA2 Encryption.**

Theorem 1. *If the encryption scheme is IND-CCA2, then the following formula is computationally sound.*

$$\mathsf{RandGen}(K) \ \wedge\ eK \sqsubseteq \hat{\phi} \ \wedge\ \mathsf{fresh}(R; \hat{\phi}, \boldsymbol{x}, x, y) \ \wedge\ \boldsymbol{x}, x, y \preccurlyeq \hat{\phi} \ \wedge\ \hat{\phi}, \boldsymbol{x}, \{x\}^R_{eK} \vdash y$$
$$\longrightarrow\ dK \sqsubseteq \hat{\phi}, \boldsymbol{x}, x \ \vee\ \hat{\phi}, \boldsymbol{x} \vdash y$$

This axiom says that if K was correctly generated, R is fresh, and y can be derived with the help of $\{x\}^R_{eK}$, then it can be derived without $\{x\}^R_{eK}$, or dK has been sent out. The secrecy axiom was essentially proven in [7], but as we need a little stronger version, we include the proof in Section 5.1. Note, that this axiom may look like it would work for CPA security, but it does not in general, as in general honest agents can be used as decryption oracles. However, for proving this axiom we only need the decryption oracle in the CCA2 game *before* the ciphertext was created by the encryption oracle, and not after.

- **Non-Malleability of CCA2 Encryption.** Suppose we have pairing as before. Let $f_1, ..., f_n$ be the rest of the non-0-arity function symbols not in $\mathcal{F}_c \cup \mathcal{F}_d$ from Section 3. Let $\mathsf{Otherf}(u)$ be a constraint meaning that u is any term that is a bunch of paired together terms all of which occur under one of the functions $f_1, \ldots, f_n$ in ϕ.

Theorem 2. *If the encryption scheme is IND-CCA2, then the following formula is computationally sound.*

$$\forall u(\mathsf{Otherf}(u) \rightarrow \hat{\phi}, u \not\vdash N) \ \wedge$$
$$\mathsf{RandGen}(N) \ \wedge\ \mathsf{RandGen}(K) \ \wedge\ eK \sqsubseteq \hat{\phi} \ \wedge\ N \sqsubseteq \hat{\phi} \ \wedge\ \boldsymbol{x} \preccurlyeq \hat{\phi} \ \wedge\ \hat{\phi}, \boldsymbol{x} \vdash y \ \wedge$$
$$\hat{\phi}, \boldsymbol{x}, dec(y, dK) \vdash N \ \wedge\ \forall x R(y = \{x\}^R_{eK} \rightarrow \{x\}^R_{eK} \not\sqsubseteq \hat{\phi}) \ \longrightarrow\ dK \sqsubseteq \hat{\phi} \ \vee\ \hat{\phi}, \boldsymbol{x} \vdash N$$

This means that if N and K were correctly generated, y can be decrypted and from the plaintext, N can be derived, but no honest agent ever produced y as an encryption, then either N can be derived without the plaintext of y, or dK has been sent out. For this, we need the full power of the CCA2 security, decryption oracle calls both before and after encryption oracle calls. The first line is necessary for making sure that function symbols other than the ones related to pairing or encryption do not interfere with our CCA2 encryption. In principle it is possible to have another encryption for example that may allow to fake encryptions of our CCA2 encryption. The condition we included ensures that the combined information contained in items occurring under the other function symbols cannot corrupt N. So if we blindly introduce a new primitive, we have to show in the course of the security proof that the first line is satisfied and the new primitive does not have a chance to interfere. However, typically, other primitives should be introduced so that they do not interfere with the non-malleability property of the encryption scheme, and if that is proven separately for all added primitives, then the first line is not needed in the protocol proof. In our NSL case, there is no first line as there are no other function symbols. The proof is presented in Section 5.2

– Special to c_i, c_r. These axioms are trivial as c_i, c_r are just ideal functions introduced for convenience to represent the agents' view of a session. (Let c be either of them):
  - c does not help the adversary: $\mathsf{RandGen}(N) \wedge \hat{\phi}, \boldsymbol{x}, c(x, y, z, w) \vdash N \rightarrow \hat{\phi}, \boldsymbol{x} \vdash N$
  - c cannot be forged and cannot be subpart of a term: $\hat{\phi}, \boldsymbol{x} \vdash c(x, y, z, w) \longrightarrow c(x, y, z, w) \sqsubseteq \hat{\phi} \vee x_1 = c(x, y, z, w) \vee \ldots \vee x_l = c(x, y, z, w)$

  - c cannot be equal to anything else: If the outermost function symbol of a term T is something different from c, then $c(x, y, z, w) \neq T$.

– **One Extra Axiom.** For the NSL protocol, we need an additional axiom, namely,

$$\mathsf{RandGen}(N) \to \neg W(\pi_2(N)).$$

That is, the second projection of a nonce can never be a name (by overwhelming probability on a non-negligible set). We assume that the implementation of the pairing is such that this condition is satisfied. If this does not hold, there is an attack which we include in Appendix A. It is very easy to ensure that an implementation satisfies this property. If the length of nonces is fixed for a given security parameter, and agents check the length of bit strings that are in the positions of nonces, in this case $\pi_1(N)$, then we can prevent $W(\pi_2(N))$ as it is easy to show that $W(\pi_2(N))$ is only possible if the length of $\pi_1(N)$ differs from the length of N with non-negligible probability. But they should be the same length as they are both nonces. To be more precise, in this case we should add in the above formula that both $W(\pi_1(N))$ and N are of nonce type (meaning they have the same computational length):

$$\mathsf{RandGen}(N) \wedge \mathtt{NonceLength}(N) \wedge \mathtt{NonceLength}(\pi_1(N)) \to \neg W(\pi_2(N)).$$

This axiom is satisfied in an implementation that checks the length of bit strings corresponding to nonces. This means that *security of the NSL protocol does not require tagging of nonces, pairs, encryptions*. The axiom is used under 2.2.2 of the proof of Proposition 1.

5 Computational Soundness of the CCA2 Axioms

Computational semantics is defined rigorously in [6]. We use that definition to show computational soundness.

5.1 Computational Soundness of the Secrecy Axiom

First we prove soundness of the secrecy axiom for CCA2 security, Theorem 1: If the encryption scheme is IND-CCA2, then the following formula is computationally sound.

$$\begin{aligned}&\mathsf{RandGen}(K) \ \wedge \ eK \sqsubseteq \hat{\phi} \ \wedge \ \mathsf{fresh}(R; \hat{\phi}, \boldsymbol{x}, x, y) \ \wedge \ \boldsymbol{x}, x, y \preccurlyeq \hat{\phi} \ \wedge \ \hat{\phi}, \boldsymbol{x}, \{x\}_{eK}^{R} \vdash y \\ &\quad \longrightarrow \ dK \sqsubseteq \hat{\phi}, \boldsymbol{x}, x \ \vee \ \hat{\phi}, \boldsymbol{x} \vdash y\end{aligned}$$

Proof. Let θ denote the formula. Suppose that it is not computationally valid. Then there is a protocol Π and an interactive turing machine (attacker) $\mathcal{M}$ such that

$$\mathcal{M}, \Pi \not\models^{c} \theta.$$

This, by the computational semantics in [6], is equivalent with that there is an S_1 computable, non-negligible set of the underlying probability field such that $\mathcal{M}, \Pi, S_1 \models^{c} \neg\theta$, which, by first-order deductions, is the same as

$$\begin{aligned}\mathcal{M}, \Pi, S_1 \models^{c} \ \exists x \boldsymbol{x} y K R \Big(&\mathsf{RandGen}(K) \ \wedge \ eK \sqsubseteq \hat{\phi} \ \wedge \ \mathsf{fresh}(R; \hat{\phi}, \boldsymbol{x}, x, y) \ \wedge \ \boldsymbol{x}, x, y \preccurlyeq \hat{\phi} \\ &\wedge \ \hat{\phi}, \boldsymbol{x}, \{x\}_{eK}^{R} \vdash y \ \wedge \ \neg(dK \sqsubseteq \hat{\phi}, \boldsymbol{x}, x) \ \wedge \ \neg(\hat{\phi}, \boldsymbol{x} \vdash y)\Big).\end{aligned}$$

This on the other hand means that there are PT machines $\mathcal{A} = (\mathcal{A}_x, \mathcal{A}_{x_1}, ..., \mathcal{A}_{x_l}, \mathcal{A}_y\mathcal{A}_K, \mathcal{A}_R)$ such that

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \mathsf{RandGen}(K)$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c eK \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \mathsf{fresh}(R, \hat{\phi}, \boldsymbol{x}, x, y)$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \boldsymbol{x}, x, y \preccurlyeq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \neg(dK \ \sqsubseteq \ \hat{\phi}, \boldsymbol{x}, x)$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x}, \{x\}_{eK}^R \vdash y$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash y).$$

We have to create an adversary $\mathcal{A}_{\mathrm{CCA2}}$ that wins the CCA2 game.

Since $\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x}, \{x\}_{eK}^R \vdash y$ holds, there is an $S_2 \subseteq S_1$ and an algorithm $\mathcal{C}$ that computes the interpretation of y from the interpretations of $\hat{\phi}$, $\boldsymbol{x}$ and $\{x\}_{eK}^R$ on S_2. Clearly,

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c \mathsf{RandGen}(K)$$

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c eK \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c \mathsf{fresh}(R; \hat{\phi}, \boldsymbol{x}, x, y)$$

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c \boldsymbol{x}, x, y \preccurlyeq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c \neg(dK \ \sqsubseteq \ \hat{\phi}, \boldsymbol{x}, x)$$

$$\mathcal{M}, \Pi, S_2, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash y).$$

It may be the case that the S_2 we have chosen depends on evaluations of τ that are determined after $\mathcal{M}$ reaches the challenge state q_c. However, clearly, if we include all possible future evaluations, the set that we receive this way, S will still be such that there is an algorithm $\mathcal{C}$ that computes the interpretation of y from the frame at the challenge state q_c, $\boldsymbol{x}$ and $\{x\}_{eK}^R$ on S. Moreover, it is easy to see that

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c \mathsf{RandGen}(K)$$

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c eK \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c \mathsf{fresh}(R; \hat{\phi}, , \boldsymbol{x}, x, y)$$

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c \boldsymbol{x}, x, y \preccurlyeq \hat{\phi}$$

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c \neg(dK \ \sqsubseteq \ \hat{\phi}, \boldsymbol{x}, x)$$

$$\mathcal{M}, \Pi, S, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash y)$$

because these are properties that depend only on conditions in the challenge stated, and not later ones.

Since $\mathcal{M}, \Pi, S, \mathcal{A} \models^c dK \ \not\sqsubseteq \ \hat{\phi}, \boldsymbol{x}, x$, the decryption key has never been sent.

We show that we can construct an algorithm $\mathcal{A}_{\mathrm{CCA2}}$ that breaks CCA2 security. Let $\mathcal{A}_\Pi$ be the protocol adversary.

- $\mathcal{A}_{\mathrm{CCA2}}$ generates computational keys that $\mathcal{A}_{\Pi}$ uses, except for the one corresponding to K.
- The encryption oracle generates a random bit b.
- The encryption oracle generates a computational key and publishes its public part. $\mathcal{A}_{\mathrm{CCA2}}$ encrypts with this key for encryptions with K, except for x.
- $\mathcal{A}_{\mathrm{CCA2}}$ simulates both the agents and $\mathcal{A}_{\Pi}$: It computes all messages that the agents output according to their algorithm, and computes all messages that $\mathcal{A}_{\Pi}$ outputs according to its algorithm. This way it builds up ϕ and the bit strings corresponding to them as well as the equations.
- Whenever a decryption with dK has to be computed, there are two possibilities:
  - If the ciphertext was created by $\mathcal{A}_{\mathrm{CCA2}}$ using the encryption algorithm, then it knows the plaintext, so it can use it without decryption.
  - If the ciphertext was created in some other way, the decryption oracle is used. This can be freely done until x occurs.

  Note, that dK is not available to the adversary.
- When $\mathcal{A}$ reaches the challenge state q_c, using $\mathcal{A}_x$, $\mathcal{A}_{\mathrm{CCA2}}$ computes the bit string for x, and submits it to the encryption oracle as well as a random bit string that has the same length as the plaintext.
- According to our definition of satisfaction the computation by $\mathcal{C}$ is based on the frame at the challenge state. We had $\mathcal{M}, \Pi, S, \mathcal{A} \models^c \mathsf{fresh}(R; \hat{\phi}, , \boldsymbol{x}, x, y)$, which means that R was honestly generated and has not been sent around, and hence R is independent of the items in ϕ. Moreover, since $\boldsymbol{x} \preccurlyeq \hat{\phi}$ and $x \preccurlyeq \hat{\phi}$ and $y \preccurlyeq \hat{\phi}$ are satisfied, R is also independent of $\boldsymbol{x}$ and x and y. Further, since we included all future random choices in S, R is also independent of S. Hence having it encrypted by the encryption oracle will not lose any information as long as the oracle encrypts the correct bit.
- The encryption oracle encrypts the interpretation of x if $b = 0$, and encrypts the random bit string if $b = 1$. It gives the result c back to $\mathcal{A}_{\mathrm{CCA2}}$.
- Run $\mathcal{C}$ on the bit string c returned by the oracle and on the bit strings of the frame.
- If, in the above
  - using c, the execution is in S and $\mathcal{A}_{\mathrm{CCA2}}$ receives the same value as $\mathcal{A}_y$ gives for y, then $\mathcal{A}_{\mathrm{CCA2}}$ returns $b_{\mathcal{A}_{\mathrm{CCA2}}} = 0$.
  - Otherwise $\mathcal{A}_{\mathrm{CCA2}}$ throws a fair coin and returns $b_{\mathcal{A}_{\mathrm{CCA2}}} = 0$ or $b_{\mathcal{A}_{\mathrm{CCA2}}} = 1$ with probability $1/2$.

We have now that

$$\mathrm{Prob}\left(b_{\mathcal{A}_{\mathrm{CCA2}}} = b \mid S \ \wedge\ b = 0\right) - 1 \tag{1}$$

(the conditional probability of $b_{\mathcal{A}_{\mathrm{CCA2}}} = b$ given S and $b = 0$) is negligible because in this case the oracle encrypts the correct string, and $\mathcal{C}$'s computations are employed on the correct bit string, so it gives the interpretation of x. Note, we also use here that S and the interpretation of R do not correlate.

On the other hand, observe that

$$\mathrm{Prob}\left(b_{\mathcal{A}_{\mathrm{CCA2}}} = b \mid S \ \wedge\ b = 1\right) - 1/2 \tag{2}$$

is also negligible. The reason is that when $b = 1$, the encryption oracle computes something that has nothing to do with the protocol and x and y. So the probability of computing x with or without the encryption in this case, is the same. But, remember, we had that $\mathcal{M}, \Pi, S, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x} \not\vdash y$. This means that y cannot be computed without the encryption anywhere and therefore the adversary's computation on the fake encryption cannot give good result by more than negligible probability. So the adversary will end up throwing a coin in this case. Putting (1) and (2) together, we have

$$\mathrm{Prob}\,(b_{\mathcal{A}_{\mathrm{CCA2}}} = b \mid S) - \frac{1}{2}$$

is non-negligible. Then, since $\mathcal{A}_{\mathrm{CCA2}}$ throws a fair coin outside S,

$$\mathrm{Prob}\,(b_{\mathcal{A}_{\mathrm{CCA2}}} = b) - \frac{1}{2}$$

is non-negligible, which means CCA2 security assumption is broken. $\square$

5.2 Computational Soundness of the Non-Malleability Axiom

We now prove the non-malleability axiom for CCA2 security, Theorem 2: If the encryption scheme is IND-CCA2, then the following formula is computationally sound.

$$\begin{aligned}&\forall u(\mathsf{Otherf}(u) \rightarrow \hat{\phi}, u \not\vdash N) \wedge \\ &\mathsf{RandGen}(N) \ \wedge\ \mathsf{RandGen}(K) \ \wedge\ eK \sqsubseteq \hat{\phi} \ \wedge\ N \sqsubseteq \hat{\phi} \ \wedge\ \boldsymbol{x} \preccurlyeq \hat{\phi} \ \wedge\ \hat{\phi}, \boldsymbol{x} \vdash y \wedge \\ &\hat{\phi}, \boldsymbol{x}, dec(y, dK) \vdash N \ \wedge\ \forall x R(y = \{x\}^R_{eK} \rightarrow \{x\}^R_{eK} \not\sqsubseteq \hat{\phi}) \ \ \longrightarrow\ \ dK \sqsubseteq \hat{\phi} \ \vee\ \hat{\phi}, \boldsymbol{x} \vdash N\end{aligned}$$

Proof. The first line just means that N was never included under anything else except for the function symbols related to pairing and encryption. So we assume this from now, and denote the rest of the formula by θ. Suppose it is not computationally valid. Just as in the proof of the secrecy axiom, it follows from the computational semantics in [6] that there is a computational structure $(\mathcal{M}, \Pi, S_1)$, with

$$\mathcal{M}, \Pi, S_1 \models^c \neg\theta.$$

By first-order deductions, this means

$$\begin{aligned}&\mathcal{M}, \Pi, S_1 \models^c \\ &\exists \boldsymbol{x} xyzKR\Big(\mathsf{RandGen}(N) \ \wedge\ \mathsf{RandGen}(K) \ \wedge\ eK \sqsubseteq \hat{\phi} \ \wedge\ N \sqsubseteq \hat{\phi} \ \wedge\ \boldsymbol{x} \preccurlyeq \hat{\phi} \ \wedge\ \hat{\phi}, \boldsymbol{x} \vdash y \wedge \\ &\quad \hat{\phi}, \boldsymbol{x}, dec(y, dK) \vdash N \ \wedge\ \forall x R(y = \{x\}^R_{eK} \rightarrow \{x\}^R_{eK} \not\sqsubseteq \hat{\phi}) \ \wedge\ \neg(dK \sqsubseteq \hat{\phi}) \ \wedge\ \neg(\hat{\phi}, \boldsymbol{x} \vdash N)\Big).\end{aligned}$$

Hence here are PPT machines $\mathcal{A} = (\mathcal{A}_{x_1}, ..., \mathcal{A}_{x_k}, \mathcal{A}_x, \mathcal{A}_y, \mathcal{A}_z, \mathcal{A}_K, \mathcal{A}_R)$ such that

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \mathsf{RandGen}(N)$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \mathsf{RandGen}(K)$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c eK \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c N \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \boldsymbol{x} \preccurlyeq \hat{\phi}$$

$$\mathcal{M}, \Pi, S_1, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x} \vdash y$$

$$\mathcal{M}, \mathit{\Pi}, S_1, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x}, dec(y, dK) \vdash N$$

$$\mathcal{M}, \mathit{\Pi}, S_1, \mathcal{A} \models^c \forall x R(y = \{x\}^R_{eK} \to \{x\}^R_{eK} \not\sqsubseteq \hat{\phi})$$

$$\mathcal{M}, \mathit{\Pi}, S_1, \mathcal{A} \models^c \neg(dK \sqsubseteq \hat{\phi})$$

$$\mathcal{M}, \mathit{\Pi}, S_1, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash N).$$

Since $\mathcal{M}, \mathit{\Pi}, S_1, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x} \vdash y$ holds, there is an $S_2 \subseteq S_1$ and an algorithm $\mathcal{C}$ that computes the interpretation of y from the interpretation of $\hat{\phi}, \boldsymbol{x}$ on S_2. Since $\mathcal{M}, \mathit{\Pi}, S_2, \mathcal{A} \models^c \hat{\phi}, \boldsymbol{x}, dec(y, dK) \vdash N$ holds, there is an $S_3 \subseteq S_2$ and an algorithm $\mathcal{C}'$ that computes the interpretation of N from the interpretations of $\hat{\phi}$ and $\boldsymbol{x}$ and $dec(y, dK)$ on S_3. Clearly, on S_3 as well, $\mathcal{C}$ computes the interpretation of y from the interpretation of $\hat{\phi}$ and $\boldsymbol{x}$.

It may be the case that the S_3 we have chosen depends on evaluations of τ that are determined after $\mathcal{M}$ reaches the challenge state q_c. However, clearly, if we include all possible future evaluations, the set that we receive this way, S will still be such that there is an algorithm $\mathcal{C}$ that computes the interpretation of y at the challenge state q_c, and $\mathcal{C}'$ that computes the interpretation of N.

Moreover,

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \mathsf{RandGen}(N)$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \mathsf{RandGen}(K)$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c eK \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c N \sqsubseteq \hat{\phi}$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \boldsymbol{x} \preccurlyeq \hat{\phi}$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \forall x R(y = \{x\}^R_{eK} \to \{x\}^R_{eK} \not\sqsubseteq \hat{\phi})$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \neg(dK \sqsubseteq \hat{\phi})$$

$$\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash N).$$

Observe first that there had to be an honest encryption in which N was sent out. The reason is the following. $\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c N \sqsubseteq \hat{\phi}$ ensures that N was sent out and $\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \neg(\hat{\phi}, \boldsymbol{x} \vdash N)$ guarantees that the adversary has not access to it. Since we assumed that there is only pairing and encryption, the only things the adversary may not have access in a term of the frame, are the ones under encryptions with honest keys. So there was at least one honest encryption of N sent out.

Secondly, observe that $\mathcal{M}, \mathit{\Pi}, S, \mathcal{A} \models^c \forall x R(y = \{x\}^R_{eK} \to \{x\}^R_{eK} \not\sqsubseteq \hat{\phi})$ means that for every $S' \subseteq S$, if $y = \{x\}^R_{eK}$ on S', then $\{x\}^R_{eK} \not\sqsubseteq \hat{\phi}$ on S'. So on any such S', y was not created by one of the agents as an honest encryption with eK.

The simplest now is to use the CCA2 definition that allows multiple agents and multiple encryption queries (this is equivalent to the standard definition). The CCA2 attacker $\mathcal{A}_{\mathrm{CCA2}}$ runs the protocol, and whenever there is an encryption by one of the honest agents with one of the honest agents' keys, the plaintext is submitted to the encryption oracle along with a randomly generated item with the same length. The encryption oracle chooses according to their internal random choice b_0 (same for all encryption oracles) that is kept the same until the end. Other encryptions are done by the CCA2 adversary. When the protocol roles are supposed to decrypt an encryption that came from the oracle, the correctly recorded plaintext is used. At the challenge state, the interpretation of y is computed with $\mathcal{C}$ and is submitted to the decryption oracle. As we have seen above, this message was not created by an honest agent, hence it was created by the adversary and so it did not come from the encryption oracle. S is checked. Outside S, the adversary throws a coin and outputs the result. Inside S, the algorithm $\mathcal{C}'$ is applied to the interpretation of the $\hat{\phi}$ and $dec(y, dK)$. Inside S, as long as the encryption oracles encrypted the correct bit string, the output of $\mathcal{C}'$ is the interpretation of N. If they encrypted the random bit string, then the output of $\mathcal{C}'$ cannot be the interpretation of N, because in this case none of the honest encryptions have anything to do with the protocol (but all information about N is under honest encryptions). So, if $\mathcal{C}'$ returns N, $\mathcal{A}_{\mathrm{CCA2}}$ outputs 0, while if it does not output N, $\mathcal{A}_{\mathrm{CCA2}}$ throws a fair coin. We have

$$\mathrm{Prob}\,(b_0 = b : b \leftarrow \mathcal{A}_{\mathrm{CCA2}}{}^{\eta} \mid S^{\eta} \wedge b_0 = 0) - 1$$

is negligible as in this case the encryption oracle encrypted the correct plaintexts. Now,

$$\mathrm{Prob}\,(b_1 =: b \leftarrow \mathcal{A}_{\mathrm{CCA2}}{}^{\eta} \mid S^{\eta} \wedge b_0 = 1) - \frac{1}{2}$$

is also negligible, because in this case, the encrypted items have nothing to do with the protocol, and so the adversary throws a coin. Since by construction we also have that

$$\mathrm{Prob}\,\big(b_1 =: b \leftarrow \mathcal{A}_{\mathrm{CCA2}}{}^{\eta} \mid (S^{\eta})^{\perp} \wedge b_0 = 1\big) - \frac{1}{2}$$

is negligible, it follows from the 3 equations above that that

$$\mathrm{Prob}\,(b_1 = b : b \leftarrow \mathcal{A}_{\mathrm{CCA2}}{}^{\eta}) - \frac{1}{2}$$

is non-negligible hence breaking CCA2 security assumption. □

6 Examples for Proving Inconsistency

Before getting into the correctness proof of the NSL protocol, we first look at three small example proofs. This way the reader will become familiar how the axioms work. Note that these derivations are *not pure first-order deductions*. Not only we use the axioms and first order deduction rules, but we also use how a symbolic execution is defined (and the transition system is not formalized in FOL).

Example 4. We start with a very trivial example. It is rather obvious that in the execution of the NSL protocol in Example 1, $\phi_2 \not\vdash A$ should be inconsistent with the axioms as A was included in ϕ_2. We can derive it the ollowing way: observe that

$$\phi_2 \equiv A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \equiv \phi_0, A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B}.$$

From the self-derivability axiom at step 0, $\phi_0, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B}, A \vdash A$. By commutativity it follows that, $\phi_0, A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash A$, which means that

$$\phi_0, A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \not\vdash A$$

is inconsistent with the axioms. So, $\mathcal{M}, \sigma \not\models A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \not\vdash A$ for any model $\mathcal{M}$, which implies that $\phi_2 \not\vdash A$ is inconsistent with the axioms.

Example 5. We can also derive, as expected, that $\phi_2 \vdash N_1$ is inconsistent with the axioms in our NSL example. This should be the case, as N_1 has only been sent out under a single good encryption. As

$$\phi_2 \equiv A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \equiv \phi_1, \{\langle N_1, A\rangle\}^{R_1}_{eK_B},$$

it is enough to show that $\phi_1, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash N_1$ is inconsistent with the axioms. Suppose that, in order to get a contradiction, this is not the case, *i.e.*,

$$\phi_1, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash N_1. \tag{3}$$

To apply the secrecy axiom consider that $\boldsymbol{x} = \langle\rangle, x = \langle N_1, A\rangle$, and $y = N_1$. Since K_B was correctly generated (appeared as a name), $\mathsf{RandGen}(K_B)$ holds. By Example 1 we have $eK_B \sqsubseteq \phi_1$. $\mathsf{fresh}(R; \phi_1, \boldsymbol{x}, x, y)$ also holds because $\mathsf{RandGen}(R_1) \wedge R_1 \not\sqsubseteq \phi_1, \langle\rangle, \langle N_1, A\rangle, N_1$. Finally, since $\boldsymbol{x} \preccurlyeq \phi_1$, $x \preccurlyeq \phi_1$ and $y \preccurlyeq \phi_1$ because none has handles, (3) holds, and $dK_B \not\sqsubseteq \phi_1, \boldsymbol{x}, x$, one may apply the secrecy axiom and get

$$\phi_1 \vdash N_1.$$

Finally, we may show that at Step 1 we have $\mathsf{fresh}(N_1; \phi_1)$, so $\mathsf{fresh}(N_1; \phi_1) \wedge \phi_1 \vdash N_1$ which contradicts the no telepathy axiom.

Example 6. From the axioms, we can also derive the increasing knowledge of an adversary, *i.e.*, for any m and x, if $\phi_m \vdash x$ is derivable from the axioms and agent checks, then $\phi_{m+1} \vdash x$ is also derivable from the axioms and agent checks. The proof is rather simple. Assume that $\phi_m \vdash x$ is derivable from the axioms and agent checks. Let t be the message sent in the $m + 1$'th step *i.e.*, $\phi_{m+1} \equiv \phi_m, t$. The increasing capabilities axiom applied to step m means $\phi_m \vdash x$ implies $\phi_m, t \vdash x$. But that is the same as

$$\phi_{m+1} \vdash x.$$

Note that from the above and from Example 4 it also follows that for any m, the axioms imply that $\phi_m \vdash A$. It is clear from Example 4, that $\phi_2 \vdash A$ follows from axioms. Then from the above, by induction, $\phi_m \vdash A$ follows from axioms. Here, we applied induction. But note, that the induction is not within FOL, we used the induction on the number of execution steps.

7 Correctness Proof of NSL

In this section we present the correctness of the NSL protocol for any (bounded) number of sessions. Namely, we show that in the symbolic execution defined above, violation of secrecy or authentication is inconsistent with the axioms. As we mentioned, we assume that agent A only executes the initiator role, and agent B only executes the responder role. But, we allow both A and B to have other sessions running with possibly corrupted agents. We start by showing that throughout the entire execution, nonces that were generated by honest initiator A and sent to honest responder B, or vice-versa, remain secret. We do this by picking any step m of the execution tree, and listing all possible execution rounds (according to the protocol roles), we show that for all possibilities, $\phi_m \not\vdash N$ together with the axioms and the agent checks imply $\phi_{m+1} \not\vdash N$. In other words, $\phi_m \not\vdash N$, the axioms and the agent checks, and $\phi_{m+1} \vdash N$ are inconsistent. Since $\phi_0 \not\vdash N$ initially holds by no-telepathy, by induction, we have $\phi_m \not\vdash N$ after any finite number of steps m. The reader can see below that the induction hypothesis is a little more complex but essentially this is what we do.

Once secrecy is proven, authentication and agreement are shown. We pick the point on the execution tree when the responder finished his task and using that we have shown that nonces remain secret, together with non-malleability we show that the initiator also had finished his task and the corresponding values that the two parties see have to match. In other words, B finished, A not finished or values don't match, and the axioms and the agent checks are inconsistent.

7.1 Secrecy

The aim of the secrecy proof is to show that nonces N sent between A and B remain secret. If N is a nonce sent by A to B means that $\exists R\big(\{N, A\}^R_{eK_B} \sqsubseteq \hat{\phi}\big)$. If B sent it to A, that means that $\exists hR\big(\{\pi_1\left(dec(h, dK_B)\right), N, B\}^R_{eK_A} \sqsubseteq \hat{\phi}\big)$. So, these nonces can be characterized by the condition

$$\begin{aligned} C[N] \;\equiv\; & \mathsf{RandGen}(N) \\ & \wedge\; \Big(\exists R.\{N, A\}^R_{eK_B} \sqsubseteq \hat{\phi} \vee \exists hR.\{\pi_1\left(dec(h, dK_B)\right), N, B\}^R_{eK_A} \sqsubseteq \hat{\phi}\Big) \end{aligned}$$

where the first exists characterizes nonces sent from A to B and the second characterizes nonces sent from B to A. Then the secrecy property we want to show is that $\forall N\big(C[N] \longrightarrow \hat{\phi} \not\vdash N\big)$. It is equivalent to show that its negation,

$$\exists N\big(C[N] \wedge \hat{\phi} \vdash N\big), \tag{4}$$

is inconsistent with the axioms and the agent checks on every possible symbolic trace.

Suppose the total length of the symbolic trace in question is n. At the end of the trace the frame ϕ contains n terms. Let us denote the frames at each node of this trace by ϕ_0, ϕ_1, ϕ_2, etc. Each frame contains one more term than the previous one. Satisfaction of $C[N]$ by this trace means that one of the terms $\{N, A\}^R_{eK_B}$ or $\{\pi_1\left(dec(h, dK_B)\right), N, B\}^R_{eK_A}$ appears in frame ϕ_n for some h, R. Let us fix such N. Furthermore, if $\boldsymbol{x}$ is a list of a finite number of nonces $\boldsymbol{x} \equiv N_1, ..., N_l$ such that they were all generated by either A or B (possibly intended to each other, possibly intended for other possible malicious agents), and they are all different from N, then we say condition $C'[\boldsymbol{x}, N]$ is satisfied. This can be written as a first order formula like

$$\begin{aligned} & C'[N_1, ..., N_l, N] \equiv \\ & \bigwedge_{i=1}^{l} \begin{pmatrix} \mathsf{RandGen}(N_i) \;\wedge\; N \neq N_i \;\wedge \\ \exists QR.\{N_i, A\}^R_{eK_Q} \sqsubseteq \hat{\phi} \;\vee\; \exists QhR.\{\pi_1\left(dec(h, dK_B)\right), N_i, B\}^R_{eK_Q} \sqsubseteq \hat{\phi} \end{pmatrix} \end{aligned}$$

We will carry out an inductive proof on the length of ϕ_l. As it turns out, in order to avoid loops in the proof, instead of (4), it is better to prove that

$$\exists N \exists \boldsymbol{x}\big(C[N] \wedge C'[\boldsymbol{x}, N] \wedge \hat{\phi}, \boldsymbol{x} \vdash N\big) \tag{5}$$

is inconsistent with the axioms and agent checks. On the symbolic trace, this means that for all l,

$$\exists N \exists \boldsymbol{x}\big(C[N] \wedge C'[\boldsymbol{x}, N] \wedge \phi_l, \boldsymbol{x} \vdash N\big)$$

is inconsistent with the axioms and agent checks. We do this by fixing an arbitrary N satisfying $C[N]$, and for this fixed N, we do an induction on the length of ϕ. Namely, we show that having fixed N, if for some $m < l$, $\exists \boldsymbol{x}\big(C'[\boldsymbol{x}, N] \wedge \phi_m, \boldsymbol{x} \vdash N\big)$ is inconsistent with the axioms and agent checks, then

$$\exists \boldsymbol{x}\big(C'[\boldsymbol{x}, N] \wedge \phi_{m+1}, \boldsymbol{x} \vdash N\big)$$

is also inconsistent with the axioms and agent checks. Then, as at $m = 0$ the statement follows from no telepathy, we are done.

Proposition 1. *In the above execution of NSL protocol, let N be such that $C[N]$ is satisfied, and let $m < l$. If for all $\boldsymbol{x}$ such that $C'[\boldsymbol{x}, N]$ holds, the axioms and agent checks imply (by FOL deduction rules) that $\phi_m, \boldsymbol{x} \not\vdash N$, then for all $\boldsymbol{x}$ such that $C'[\boldsymbol{x}, N]$ holds, the axioms and agent checks imply (by FOL deduction rules) that $\phi_{m+1}, \boldsymbol{x} \not\vdash N$ holds.*

Proof. Suppose in order to get a contradiction that the claim is not true. That is, let us assume that there is a finite set of nonces $\boldsymbol{x} \equiv N_1, ...N_l$ such that $C'[\boldsymbol{x}, N]$ and

$$\phi_{m+1}, N_1, ..., N_l \vdash N$$

is satisfied in some semantics. We will show show that this, together with the honest agent tests and the axioms, imply that for some nonces $x' \equiv N_1', ...N_{l'}'$ with $C'[\boldsymbol{x}', N]$,

$$\phi_m, N_1', ..., N_{l'}' \vdash N.$$

Let t be the last term in ϕ_{m+1}, that is, $\phi_{m+1} \equiv \phi_m, t$ (t was sent either by A or B). Suppose, in order to get a contradiction, that $\phi_m, t, N_1, ..., N_l \vdash N$. By commutativity, we get

$$\phi_m, N_1, ..., N_l, t \vdash N. \tag{6}$$

We divide the proof in two cases: when t was sent by A and when t was sent by B.

1.) Assume that t was sent by A. According to the role of A, for some N_1', R_1, h_3, R_3, Q,

1. $t \equiv \{N_1', A\}_{eK_Q}^{R_1}$, or
2. $t \equiv \{\pi_1\left(\pi_2\left(dec(h_3, dK_A)\right)\right)\}_{eK_Q}^{R_3}$, or
3. $t \equiv c_i\left(A, Q, N_1, \pi_1\left(\pi_2\left(dec(h_3, dK_A)\right)\right)\right)$.

1.1.) $t \equiv \{N_1', A\}_{eK_Q}^{R_1}$. Since A is following the initiator role honestly, A generated N_1' earlier.

1.1.1.) If $N_1' \equiv N$, then $C[N_1']$ is satisfied (because $C[N]$ was assumed to be satisfied). By definition of C that means that N_1' was sent in a message to B. As A does only the initiator role and nothing else, each message by A that looks like $\{N_1', A\}_{eK_Q}^{R_1}$ has a freshly generated nonce each time. So if $N' \equiv N$, that means that the messages themselves must also be the same and so $\{N_1', A\}_{eK_Q}^{R_1} \equiv \{N, A\}_{eK_B}^{R_1}$, which implies that $Q \equiv B$. Applying to our hypothesis (6) we get

$$\phi_m, N_1, ..., N_l, \{N_1', A\}_{eK_B}^{R_1} \vdash N.$$

As A was sent out in ϕ_1 we have $\phi_m \vdash A$ (Example 6), and by an argument similar to the one in Example 5, we get $\phi_m, N_1, ..., N_l, \{N_1', A\}_{eK_B}^{R_1} \vdash \langle N, A\rangle$.

Since for $x \equiv N_1, ..., N_l$ the predicate $x \preccurlyeq \phi_m$ holds, as the entries of x are all names, $dK_B \not\sqsubseteq \phi_m, N_1, ..., N_l, N_1', A$, and $\mathsf{fresh}(R_1; \phi_m, N_1, ..., N_l, N_1', A)$ also hold, the secrecy axiom implies that

$$\phi_m, N_1, ..., N_l \vdash N,$$

which is exactly what we had to show.

1.1.2.) Let now $N_1' \not\equiv N$. By the functions axiom, we have

$$\phi_m, N_1, ..., N_l, N_1', A, eK_Q, R_1 \vdash \{N_1', A\}_{eK_Q}^{R_1},$$

that together with hypothesis (6) and transitivity implies

$$\phi_m, N_1, ..., N_l, N_1', A, eK_Q, R_1 \vdash N.$$

Since R_1 is fresh and different from N, by the fresh items are independent axiom, we obtain $\phi_m, N_1, ..., N_l, N_1', A, eK_Q \vdash N$.

Since eK_Q is revealed explicitly at the beginning, we have $\phi_m, N_1, ..., N_l, N_1', A \vdash eK_Q$, and hence, by the transitivity axiom, $\phi_m, N_1, ..., N_l, N_1', A \vdash N$. Applying the same reasoning to A we get

$$\phi_m, N_1, ..., N_l, N_1' \vdash N.$$

By hypothesis N_1' is not N and so $C'[\boldsymbol{x}', N]$ is satisfied for $\boldsymbol{x}' \equiv N_1, ..., N_l, N_1'$ that is exactly what we wanted.

1.2.) $t \equiv \{\pi_1(\pi_2(dec(h_3, dK_A)))\}_{eK_Q}^{R_3}$. By the definition of the role followed by A, we have that $\pi_2(\pi_2(dec(h_3, dK_A))) = Q$. Applying transitivity and function axioms to (6)

$$\phi_m, N_1, ..., N_l, dec(h_3, dK_A), eK_Q, R_3 \vdash N.$$

Since eK_Q was revealed at the beginning, and R_3 is fresh, by Example 6 and the fresh items are independent axiom we get

$$\phi_m, N_1, ..., N_l, dec(h_3, dK_A) \vdash N.$$

By the non-malleability axiom (with $\boldsymbol{x} = N_1, ..., N_l, y = h_3$), we get that either

$$\exists x R(\{x\}_{eK_A}^R = h_3 \wedge \{x\}_{eK_A}^R \sqsubseteq \phi_m) \quad \text{or} \quad \phi_m, N_1, ..., N_l \vdash N.$$

1.2.1.) If $\phi_m, N_1, ..., N_l \vdash N$ we are done.

1.2.2.) Suppose now $\exists x R(\{x\}_{eK_A}^R = h_3 \wedge \{x\}_{eK_A}^R \sqsubseteq \phi_m)$. Then $\{x\}_{eK_A}^R$ has been sent out. Since A never encrypts with its own key, it had to be B who sent it out (as the only two honest agents are A and B). Therefore, as B follows its responder role,

$$\{x\}_{eK_A}^R \equiv \{\pi_1(dec(h_2, dK_B)), N_2', B\}_{eK_A}^R$$

for some N_2'. Therefore, $dec(h_3, dK_A) = \langle \pi_1(dec(h_2, dK_B)), N_2', B\rangle$. But then

$$\pi_1(\pi_2(dec(h_3, dK_A))) = N_2' \quad \text{and} \quad \pi_2(\pi_2(dec(h_3, dK_A))) = B.$$

As we also had that $\pi_2(\pi_2(dec(h_3, dK_A))) = Q$, we get $Q = B$. Substituting the appropriate terms we have $t \equiv \{\pi_1(\pi_2(dec(h_3, dK_A)))\}_{eK_Q}^{R_3} = \{N_2'\}_{eK_B}^{R_3}$. Then, by the congruence of equality with respect to $\vdash$,

$$\phi_m, N_1, ..., N_l, \{N_2'\}_{eK_B}^{R_3} \vdash N,$$

and by the secrecy axiom, as R_3 is fresh and as $\boldsymbol{x} \preccurlyeq \phi_m$,

$$\phi_m, N_1, ..., N_l \vdash N,$$

which is what we wanted.

1.3.) If $t \equiv c_i\big(A, Q, N_1, \pi_1\left(\pi_2\left(dec(h_3, dK_A)\right)\right)\big)$, then by the c is not helpful axiom, we have $\phi_m, N_1, ..., N_l, t \vdash N$.

2.) Assume now that t was sent by B. By to the role of B, for some h_2, N_2', R_2,

1. $t \equiv \{\pi_1\left(dec(h_2, dK_B)\right), N_2', B\}^{R_2}_{eK_{\pi_2\left(dec(h_2,dK_B)\right)}}$, or
2. $t \equiv c_r(\pi_2\left(dec(h_2, dK_B)\right), B, \pi_1\left(dec(h_2, dK_B)\right), N_2)$.

(and $W(\pi_2\left(dec(h_2, dK_B)\right))$ holds.) Just as in the previous case, c_r does not help the adversary to get N, so we can assume $t \equiv \{\pi_1\left(dec(h_2, dK_B)\right), N_2', B\}^{R_2}_{eK_{\pi_2\left(dec(h_2,dK_B)\right)}}$.

2.1.) If $N_2' \equiv N$, then, since $C[N]$ holds, $C[N_2']$ also holds, so as N_2' was generated by B, it was sent to A. And since B does not do anything other than executing responder sessions, and N is only generated in one session,

$$\{\pi_1\left(dec(h_2, dK_B)\right), N_2', B\}^{R_2}_{eK_{\pi_2\left(dec(h_2,dK_B)\right)}} \equiv \{\pi_1\left(dec(h_2, dK_B)\right), N, B\}^{R_2}_{eK_A},$$

But substituting in (6) we get $\phi_m, N_1, ..., N_l, \{\pi_1\left(dec(h_2, dK_B)\right), N, B\}^{R_2}_{eK_A} \vdash N$, which, by the secrecy axiom implies

$$\phi_m, N_1, ..., N_l \vdash N,$$

which is what we wanted.

2.2.) If $N_2' \not\equiv N$, by the transitivity and the functions derivability axioms, we have

$$\phi_m, N_1, \ldots, N_l, \pi_1\left(dec(h_2, dK_B)\right), N_2', B, eK_{\pi_2(dec(h_2,dK_B))}, R_2 \vdash N.$$

By the independence axiom (and commutativity), as N_2' and R_2 are fresh, we can drop them, and since B and $eK_{\pi_2(dec(h_2,dK_B))}$ were published at the beginning, so we can drop them too obtaining

$$\phi_m, N_1, \ldots, N_l, \pi_1\left(dec(h_2, dK_B)\right) \vdash N.$$

By the transitivity and the functions derivability axioms, we have

$$\phi_m, N_1, \ldots, N_l, dec(h_2, dK_B) \vdash N. \tag{7}$$

Applying non-malleability axiom to (7) we have either

$$\phi_m, N_1, \ldots, N_l \vdash N \quad \text{or} \quad \exists x R(h_2 = \{x\}^R_{eK_B} \wedge \{x\}^R_{eK_B} \sqsubseteq \phi_m).$$

In the first case, we complete our proof. In the second case, since B does not encrypt messages with the key eK_B, h_2 is sent by A. Then, we have either

$$h_2 = \{x\}^R_{eK_B} \equiv \{N_1', A\}^{R_1}_{eK_Q} \quad \text{or} \quad h_2 = \{x\}^R_{eK_B} \equiv \{\pi_1\left(\pi_2\left(dec(h_3, dK_A)\right)\right)\}^{R_3}_{eK_Q}$$

for some Q, N_1', R_1, R_3, and h_3 with $Q \equiv B$.

2.2.1.) If $\{x\}^R_{eK_B} \equiv \{N_1', A\}^{R_1}_{eK_B}$, then $N_1' \equiv \pi_1\left(dec(h_2, dK_B)\right)$ hence

$$t \equiv \{N_1', N_2', B\}^R_{eK_A}.$$

Then, the secrecy axiom applied to $\phi_m, N_1, ..., N_l, \{N_1', N_2', B\}_{eK_A}^R \vdash N$ implies

$$\phi_m, N_1, ..., N_l \vdash N$$

2.2.2.) If $\{x\}_{eK_B}^R \equiv \{\pi_1(\pi_2(dec(h_3, dK_A)))\}_{eK_B}^{R_3}$, then

$$dec(h_2, dK_B) = \pi_1(\pi_2(dec(h_3, dK_A))).$$

Substituting in Equation 7, we get $\phi_m, N_1, \ldots, N_l, \pi_1(\pi_2(dec(h_3, dK_A))) \vdash N$, hence

$$\phi_m, N_1, \ldots, N_l, dec(h_3, dK_A) \vdash N.$$

Similarly to the previous case, applying non-malleability implies that either

$$\phi_m, N_1, ..., N_l \vdash N \quad \text{or} \quad \exists x'R'.(h_3 = \{x'\}_{eK_A}^{R'} \wedge \{x'\}_{eK_A}^{R'} \sqsubseteq \phi_m).$$

In the first case, we complete the proof. If $\exists x'R'.(h_3 = \{x'\}_{eK_A}^{R'} \wedge \{x'\}_{eK_A}^{R'} \sqsubseteq \phi_m)$, then, since only B encrypts with eK_A, and since it follows its responder role, we have

$$h_3 = \{x'\}_{eK_A}^{R'} \equiv \{\pi_1(dec(h_2', dK_B)), N_2'', B\}_{eK_{\pi_2(dec(h_2', dK_B))}}^{R_2'}.$$

for some $\mathsf{RandGen}(N_2'')$. Therefore we have $\pi_1(\pi_2(dec(h_3, dK_A))) \equiv N_2''$, that is, $x \equiv N_2''$. But, we also had that $x = dec(h_2, dK_B)$, and the beginning of 2.), said $W(\pi_2(dec(h_2, dK_B)))$. Putting these three together, we get

$$W(\pi_2(N_2'')),$$

which contradicts our necessary axiom for the NSL, $\mathsf{RandGen}(N_2'') \to \neg W(\pi_2(N_2''))$. $\square$

We still have to show that $\exists N \exists \boldsymbol{x}\big(C[N] \wedge C'[\boldsymbol{x}, N] \wedge \phi_0, \boldsymbol{x} \vdash N\big)$ is inconsistent with the axioms. Let $C[N]$ and $C'[\boldsymbol{x}, N]$ hold for N and $\boldsymbol{x} \equiv N_1, ..., N_l$. At step 0, N, $N_1, ..., N_l$ are still fresh (remember, we assumed for simplicity that everything was generated upfront, and clearly, these nonces have not been sent), so by the no telepathy axiom, $\phi_0 \not\vdash N$, and then by the independence of fresh items, $\phi_0, N_1 \vdash N$. Then again by the independence of fresh items, $\phi_0, N_1, N_2 \not\vdash N$, etc. So

$$\phi_0, N_1, ..., N_l \not\vdash N$$

holds, meaning that $\exists N \exists \boldsymbol{x}\big(C[N] \wedge C'[\boldsymbol{x}, N] \wedge \phi_0, \boldsymbol{x} \vdash N\big)$ is indeed inconsistent. Then the induction step in Proposition 1 proves that this property always holds. In particular, we have the following theorem.

Theorem 3 (Secrecy). *Consider a symbolic execution of the NSL protocol, with an arbitrary number of possible dishonest participants and two honest participants A, B that follow the initiator and responder roles correspondingly, and that only execute these roles in each of their bounded number of sessions. Further, consider the convention $\langle x, y, z\rangle \equiv \langle x, \langle y, z\rangle\rangle$.*

Our axioms together with the agent checks and $\mathsf{RandGen}(N) \to \neg W(\pi_2(N))$ imply that for all n and for any nonce N that was either generated by A and sent to B, or vice versa, $\phi_n \not\vdash N$.

The above Theorem states that secrecy of nonces satisfying $C[N]$ is not broken. That is, nonces that were generated by A or B and intended to be sent between each other, remain secret. Taking $\boldsymbol{x}$ to be the empty list, the formula $\exists N\big(C[N] \wedge \hat{\phi} \vdash N\big)$, together with the axioms and the agent checks, and $\mathsf{RandGen}(N) \to \neg W(\pi_2(N))$ are inconsistent on any symbolic trace.

7.2 Agreement and Authentication

We now prove agreement from the responder's viewpoint. That is, we will show that

$$\begin{array}{l} \mathit{Resp}^B_{NSL}[B, i', N_2, h_2, h_4, R_2] \text{ AND} \\ \pi_2\left(\mathit{dec}(h_2, dK_B)\right) = A \end{array} \Longrightarrow \begin{array}{l} \text{EXIST } i, N_1, h_1, h_3, R_1, R_3 \text{ SUCH THAT} \\ \mathit{Init}^A_{NSL}[A, i, B, N_1, h_1, h_3, R_1, R_3] \text{ AND} \\ \mathit{dec}(h_2, dK_B) = \langle N_1, A\rangle \text{ AND} \\ \mathit{dec}(h_3, dK_A) = \langle N_1, N_2, B\rangle \text{ AND} \\ \mathit{dec}(h_4, dK_B) = N_2 \end{array}$$

Where by the implication sign we mean that the agent checks, and axioms imply this. We can also write this within our syntax:

$$\begin{array}{l} A = \pi_2\left(\mathit{dec}(h_2, dK_B)\right) \wedge \\ N_1 = \pi_1\left(\mathit{dec}(h_2, dK_B)\right) \wedge \\ c_r(A, B, N_1, N_2) \sqsubseteq \hat{\phi} \wedge \end{array} \longrightarrow \exists h_3. \begin{pmatrix} c_i(A, B, N_1, N_2) \sqsubseteq \hat{\phi} \wedge \\ N_2 = \pi_1\left(\pi_2\left(\mathit{dec}(h_3, dK_A)\right)\right) \end{pmatrix}$$

What we have to prove is that the negation of this is inconsistent with the axioms and agent checks. But for that it is sufficient to show that the agent checks and axioms and the premise of the formula imply the conclusion of this formula.

Theorem 4 (Agreement and Authentication).
Consider a symbolic execution of the NSL protocol, with an arbitrary number of possible dishonest participants and two honest participants A, B that follow the initiator and responder roles correspondingly, and that only execute these roles in each of their bounded number of sessions. Further, consider the convention $\langle x, y, z\rangle \equiv \langle x, \langle y, z\rangle\rangle$.

Our axioms together with the agent checks and $\mathsf{RandGen}(N) \to \neg W(\pi_2(N))$ *are inconsistent with the negation of the formula*

$$\begin{array}{l} c_r(\pi_2\left(\mathit{dec}(h_2, dK_B)\right), B, \pi_1\left(\mathit{dec}(h_2, dK_B)\right), N_2) \sqsubseteq \hat{\phi} \;\wedge\; A = \pi_2\left(\mathit{dec}(h_2, dK_B)\right) \\ \quad \longrightarrow \exists N_1 h_3. \begin{pmatrix} c_i(A, B, N_1, \pi_1\left(\pi_2\left(\mathit{dec}(h_3, dK_A)\right)\right)) \sqsubseteq \hat{\phi} \wedge \\ N_2 = \pi_1\left(\pi_2\left(\mathit{dec}(h_3, dK_A)\right)\right) \;\wedge\; N_1 = \pi_1\left(\mathit{dec}(h_2, dK_B)\right) \end{pmatrix} \end{array}$$

Proof. $c_r(\pi_2\left(\mathit{dec}(h_2, dK_B)\right), B, \pi_1\left(\mathit{dec}(h_2, dK_B)\right), N_2) \sqsubseteq \hat{\phi}$ means (according to the role of B) that $\mathit{Resp}^B_{NSL}[B, i', N_2, h_2, h_4, R_2]$ was carried out and so we have

$$\{\pi_1\left(\mathit{dec}(h_2, dK_B)\right), N_2, B\}^{R_2}_{eK_A} \sqsubseteq \phi_m \tag{8}$$

(for m step, when it is sent), and

$$\mathit{dec}(h_4, dK_B) = N_2.$$

Applying the self derivability axiom we have $\phi_m, \mathit{dec}(h_4, dK_B) \vdash N_2$. and by non-malleability we have either

$$\phi_m \vdash N_2 \quad \text{or} \quad \exists x R.(h_4 = \{x\}^R_{eK_B} \wedge \{x\}^R_{eK_B} \sqsubseteq \phi_m).$$

The first case is impossible by Theorem 3. In the second case decrypting h_4 we have

$$x = N_2.$$

Since B does not send messages encrypted with eK_B, $h_4 = \{N_2\}^R_{eK_B}$ is sent by A in some session i.

1.) Case $\{x\}_{eK_B}^{R} \equiv \{N_1, A\}_{eK_Q}^{R_1}$ for some N_1 and R_1: In this case we get $x \equiv \langle N_1, A\rangle$, and so $N_2 = \langle N_1, A\rangle$. This implies by the self-derivability that for any m', $\phi_{m'}, N_1, A \vdash N_2$, and since A is public, implies

$$\phi_{m'}, N_1 \vdash N_2.$$

This is true for all m', so it is also true for the one when N_1 or N_2 is fresh. But, that contradicts either the freshness or the no telepathy axiom. If N_1 is fresh, by the freshness axiom we get $\phi_{m'} \vdash N_2$ that is impossible by Theorem 3. If N_2 is fresh, the no telepathy axiom is violated. So the assumption in 1.) is not possible.

2.) Case $\{x\}_{eK_B}^{R} \equiv \{\pi_1(\pi_2(\mathit{dec}(h_3, dK_A)))\}_{eK_Q}^{R}$: Since A follows the initiator role, there exists N_1 honestly generated by A and h_3 such that

$$\pi_1(\mathit{dec}(h_3, dK_A)) = N_1 \quad \text{and} \quad \pi_2(\pi_2(\mathit{dec}(h_3, dK_A))) = B \tag{9}$$

That is, session i of A is with agent B. Since $x \equiv \pi_1(\pi_2(\mathit{dec}(h_3, dK_A)))$ and $x = N_2$, together with (9) we have

$$\pi_1(\pi_2(\mathit{dec}(h_3, dK_A))) = N_2 \quad \text{and} \quad \mathit{dec}(h_3, dK_A) = \langle N_1, N_2, B\rangle. \tag{10}$$

So we have $\mathit{Init}_{NSL}^{A}[A, i, B, N_1, h_1, h_3, R_1, R_3]$. The only thing left to be proven is

$$\mathit{dec}(h_2, dK_B) = \langle N_1, A\rangle.$$

Applying the self derivability axiom to (9) we have $\phi_{m''}, \pi_1(\mathit{dec}(h_3, dK_A)) \vdash N_1$ for any m'', and by the functions derivability and transitivity axioms, we have

$$\phi_{m''}, \mathit{dec}(h_3, dK_A) \vdash N_1.$$

Applying the non-mallability axiom, we have

$$\phi_{m''} \vdash N_1 \quad \text{or} \quad \exists x'R'.(h_3 = \{x'\}_{eK_A}^{R'} \wedge \{x'\}_{eK_A}^{R'} \sqsubseteq \phi_{m''})$$

and the first is not possible by Theorem 3 as N_1 was generated by A in session i in which the intended party is B as we have shown. Then, we have

$$\mathit{dec}(h_3, dK_A) = x'. \tag{11}$$

Since A does not encrypt messages with eK_A, $\{x'\}_{eK_A}^{R'}$ was sent by B. Then we have

$$\{x'\}_{eK_A}^{R'} = \{\pi_1(\mathit{dec}(h_2', dK_B)), N_2', B\}_{eK_A}^{R_2'} \tag{12}$$

for some h_2', N_2', and R_2'. From (9), (11) and (12), we get

$$N_1 = \pi_1(\mathit{dec}(h_3, dK_A)) = \pi_1(x') = \pi_1(\mathit{dec}(h_2', dK_B)), \tag{13}$$

and from (10), (11) and (12), we get

$$N_2 = \pi_1(\pi_2(\mathit{dec}(h_3, dK_A))) = \pi_1(\pi_2(x')) = N_2'.$$

Since B, according to its role, always generates a new nonce before sending its message, N_2 is not used in more than one session. On the other hand, we have already concluded that in the session where N_2 is sent, (8), the message is $\{\pi_1 (dec(h_2, dK_B)), N_2, B\}^{R_2}_{eK_A}$, therefore by (12),

$$\{\pi_1 (dec(h'_2, dK_B)), N'_2, B\}^{R'_2}_{eK_A} \equiv \{\pi_1 (dec(h_2, dK_B)), N_2, B\}^{R_2}_{eK_A},$$

and so $h'_2 \equiv h_2$, which means by (13)

$$\pi_1 (dec(h_2, eK_A)) = N_1.$$

Putting all these together, we have that there exist $i, N_1, h_1, h_3, R_1, R_3$ such that

$$\begin{array}{l} Init^A_{NSL}[A, i, B, N_1, h_1, h_3, R_1, R_3] \quad \text{AND} \\ dec(h_2, dK_B) = \langle N_1, A\rangle \quad \text{AND} \\ dec(h_3, dK_A) = \langle N_1, N_2, B\rangle \quad \text{AND} \\ dec(h_4, dK_B) = N_2 \end{array}$$

which immediately implies the intended property. □

8 Conclusion and Future Work

In this paper we illustrated that the framework proposed by Bana and Comon-Lundh [7], where one does not define explicitly the Dolev-Yao adversarial capabilities but rather the limitations (axioms) on these capabilities, is suitable and powerful enough to prove correctness of security protocols. The proofs with this technique are computationally sound without the need of any further assumptions such as no bad keys, etc that are otherwise usually assumed in other literature.

We presented a modular set of axioms that are computationally sound for implementations using CCA2 secure encryption. Using the axioms together with a minimal parsing assumption, we were able to perform an inductive proof to show both secrecy and agreement of the NSL protocol. Applying the main theorem of [7] we obtain that for any implementation satisfying CCA2 security and the parsing assumption, there is no computational adversary that can violate secrecy or authentication except with negligible probability. We also believe the axioms of secrecy and non-malleability constitute a sufficient abstraction of CCA2 security to prove correctness of protocols other than NSL.

As other current techniques have problems incorporating dynamic corruption, it is worth noting, that our technique works even if the protocol allows the release of a decryption key of A or B at some time. Secrecy can still be proven until that point, and authentication that was carried out earlier can be verified even if the decryption key is later released.

The proof we presented in this paper was done by hand but we believe that automation is possible and is left for future work.

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, January 2002.
2. P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness and completeness of formal encryption: the cases of key-cycles and partial information leakage. *Journal of Computer Security*, 17(5):737–797, 2009.

3. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *CCS'03*, 2003.
4. M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (rsim) framework for asynchronous systems. *Information and Computation*, 205(12), 2007.
5. G. Bana, P. Adão, and H. Sakurada. Computationally sound verification of the NSL protocol via computationally complete symbolic attacker—Long version, 2012. http://web.ist.utl.pt/pedro.adao/pubs/drafts/nsl-long.pdf.
6. G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. Available at IACR ePrint Archive, Report 2012/019.
7. G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In *Proceedings of POST'12*, LNCS, 2012.
8. G. Bana, K. Hasebe, and M. Okada. Computational semantics for basic protocol logic - a stochastic approach. In *ASIAN'07*, volume 4846 of *LNCS*, pages 86–94. Springer, 2007.
9. G. Bana, K. Hasebe, and M. Okada. Secrecy-oriented first-order logical analysis of cryptographic protocols, 2010. http://eprint.iacr.org/2010/080.
10. G. Barthe, B. Grégoire, and S. Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *POPL'09*, pages 90–101. ACM, 2009.
11. B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, 2008.
12. H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *CCS'08*, 2008.
13. H. Comon-Lundh and V. Cortier. How to prove security of communication protocols? A discussion on the soundness of formal models w.r.t. computational ones. In *STACS'11*, volume 9 of *Leibniz International Proceedings in Informatics*, pages 29–44, March 2011.
14. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *ESOP'05*, volume 3444 of *LNCS*, pages 157–171, 2005.
15. A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *ICALP'05*, volume 3580 of *LNCS*, pages 16–29. Springer, 2005.
16. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004.

A An Attack on NSL

If we assume that $\mathsf{RandGen}(N) \wedge W(\pi_2(N))$ is computationally satisfiable, then we have the following computational attack on the NSL protocol. $\mathsf{RandGen}(N) \wedge W(\pi_2(N))$ is the same as saying that with non-negligible probability, it is possible to choose a name (bit-string) Q for an agent such that for the output N of some honest nonce generation, there is a bit-string n such with $\langle n, Q\rangle = N$. To show that this is not at all unrealistic, suppose the pairing $\langle \cdot, \cdot \rangle$ is concatenation, and the length of agent names does not depend on the security parameter, say always 8 bits. Then for any name Q, n can be chosen with $\langle n, Q\rangle = N$ as long as the last four digits of N equals Q, which, if N is evenly generated, is of just $1/2^8$, a non-negligible probability. So this situation is realistic. Now, the attack is the following, it needs two sessions:

1. The adversary choses a name Q as above.
2. The adversary catches the last message $\{N_2\}_B$ in a session sent by A to B, two honest agents.
3. The adversary, acting as agent Q initiates a new session with B, sending $\{N_2\}_B$ to it.
4. Since B thinks this is a new session with Q, it will parse the message according to its role, namely as $\{N_1', Q\}_B$. This will succeed as long as there is an n with $\langle n, Q\rangle = N_2$, that is, it will succeed with non-negligible probability.

5. B generates a new nonce, N_2', and sends $\{n, N_2', B\}_Q$ to Q.
6. The adversary Q decrypts $\{n, N_2', B\}_Q$, reads n, and computes $N_2 = \langle n, Q\rangle$. The secrecy of N_2 is hence broken.

So, we can conclude that if $\langle n, Q\rangle = N$ is possible computationally with non-negligible probability, then the protocol fails. In such case, trace-lifting soundness proofs fail as a bit string can be understood both as $\langle n, Q\rangle$ and as N.

Notice, that this attack is not a usual type-flaw attack, because even if type-flaw attacks are allowed, honestly generated nonces are normally considered atomic.

Clearly, if the implementation of the protocol is such that B always checks the length of n, then this attack is not possible. But, it has to be made sure, that the implementation satisfies the $\mathsf{RandGen}(N) \to \neg W(\pi_2(N))$ property.