Computationally Complete Symbolic Attacker in Action

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

1 **MSR-INRIA Joint Centre, Palaiseau, 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, and two specific axioms that are sound for CCA2 secure encryptions—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. The axioms are entirely modular, not particular to the NSL protocol.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Security Protocols, Symbolic Adversary, Computational Soundness

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.546

1 Introduction

Research into computational soundness of symbolic verification of security protocols started with Abadi and Rogaway [1], followed by many others for passive [12, 10, 2] and active adversaries. The aim is always that symbolic proofs imply computational security. Works concerning active adversaries can be divided into two groups. Works in one [3, 16, 4, 14] 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 either. The other group aims to work directly in the computational model [18, 8, 13, 11]. This paper focuses on the first.

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 large sets of restrictive assumptions on computational implementations. Typically they assume that no key cycles can ever be created, that bit strings can be unambiguously parsed into terms, that there is no dynamic corruption, that keys are certified

[*] Partially supported by the ANR project ProSe and FCT project ComFormCrypt PTDC/EIA-CCO/113033/2009.
[†] Partially supported by FCT projects ComFormCrypt PTDC/EIA-CCO/113033/2009 and PEst-OE/EEI/LA0008/2011.

© Gergei Bana,Pedro Adão,Hideki Sakurada;
licensed under Creative Commons License NC-ND

(badly generated keys cannot be used), etc. These assumptions as well as reasons why they are not realistic are discussed in [15].

Recently, Bana and Comon-Lundh presented in [7] (and improved in [6]) a new technique to define symbolic attackers. They called this new symbolic adversary *computationally complete symbolic adversary*, as it is capable of doing everything that a computational adversary is capable of without any restrictions. Instead of listing every kind of move a symbolic adversary is allowed to do as usual with Dolev-Yao adversaries, a few rules (axioms) are listed that the symbolic adversary is not allowed to violate. That is, the symbolic adversary is allowed to do everything that is consistent with these axioms. The axioms are first-order formulas expressing properties that hold computationally for PPT adversaries and for the computational implementations of cryptographic primitives. 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 (that is, implementations for which the axioms are computationally sound), 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).

In this paper, we show the technique of Bana and Comon-Lundh can indeed be used for protocol verification. 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 proven 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. 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 as any term that was created by an honest agent or the adversary may *a priori* be wrongly parsed by another honest agent. The fact that they are not wrongly parsed had to be derived in the protocol proof from our axioms, that is, from reasons other than tagging. 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 have been quite short.

In [20] the author provides a direct computational proof that the NSL protocol is secure if the encryption scheme used is IND-CCA. His proof also uses bounded number of sessions, but implicitly assumes some level of unambiguous parsing (hence did not find the attack presented here). However, it allows agents to run both roles (but not against themselves).

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 computational sound axioms that are sufficient to show both secrecy and authentication of the NSL protocol. In Section 5, we show our new computational attack to the NSL. In Section 6, we show a few simple examples of how inconsistency of formulas can be proven in the framework. In Section 7, we prove that no symbolic adversary compliant with the presented axioms can violate secrecy or authentication of the NSL protocol. In Section 8, we summarize our results and present directions for future work.

2 Symbolic Execution and Properties

2.1 Terms, Frames, and Formulas

Terms are built from a set of function symbols $\mathcal{F}$ containing a countable set of *names* $\mathcal{N}$, a countable set of *handles* $\mathcal{H}$, which are of 0-arity, and a finite number of higher arity symbols. Names denote items honestly generated by agents, while handles denote inputs of the adversary. Let $\mathcal{X}$ be an infinite 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 $t_1, \ldots, t_n$.

Let $\mathcal{P}$ be a set of predicate symbols over terms. $\mathcal{P}$ contains the binary predicate $=$ used as $t_1 = t_2$, and a family of $n+1$-arity predicates $\vdash_n$ used as $t_1, ..., t_n \vdash t$, intended to model the adversary's capability to derive something. We drop the index n for readability. We *allow any FOL interpretation* of the predicates 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. We only require that $\mathcal{M}$ interprets terms and predicates such that $=$ is interpreted as the equality in the underlying domain $D_{\mathcal{M}}$. Given an assignment 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 $\mathcal{F}$ and $\mathcal{P}$, and any assignment σ of the free variables of θ in the domain of $\mathcal{M}$, the satisfaction relation $\mathcal{M}, \sigma \models \theta$, is defined as usual in FOL.

2.2 Symbolic Execution of a Protocol

▶ **Definition 2.1.** A *symbolic state* of the network consists of:
- a control state $q \in Q$ together with a sequence of names $n_1, \ldots, n_k$ (so far generated)
- 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

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

where, 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 $\sigma_{i+1} = \{x_1 \mapsto h_1, \ldots, x_{i+1} \mapsto h_{i+1}\}$ 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 2.2.** Given an interpretation $\mathcal{M}$, a transition sequence of protocol 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 assume that all honestly generated items (nonces, random inputs of encryptions, etc) are generated upfront. This is possible as we assumed bounded number of sessions. We always set $\phi_0 = \nu\overline{n}()$, where $\overline{n}$ are the honestly generated items. ϕ_1 contains the output of the initialization, that is, the names and the public keys.

2.3 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 $\hat{\phi}, x_1, ..., x_m \vdash x$ was defined in [7] and improved in [6]. 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 various *constraints*: $\mathsf{Handle}(h)$ 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); $x \sqsubseteq \vec{x}$ means x is part of $\vec{x}$. $dK \sqsubseteq_{\not d} \hat{\phi}$ means dK occurs other than in a decryption position $dec(\ ,dK)$ in ϕ, and $dK \sqsubseteq_{\not d} \vec{x}$ is analogous. Let us introduce the following abbreviations:

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

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: $\mathcal{M}, (q, \langle h_1, \ldots, h_m \rangle, \overline{n}, \phi_m, \Theta) \models \theta \quad$ iff $\quad \mathcal{M}, \phi_m, \overline{n} \models \theta$.

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. $A \to B \ : \ \{N_2\}_{eK_B}$

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 set of constructors $\mathcal{F}_c = \{\{\_\}^{\_}_{\_}, \langle \_, \_ \rangle, e\_, d\_, K\_\}$, and the set of destructors $\mathcal{F}_d = \{dec(\_, \_), \pi_1(\_), \pi_2(\_)\}$, with the following equations:

- Decryption of an encryption returns the plaintext: $dec(\{x\}^R_{eK}, dK) = x$
- First projection of pairing: $\pi_1(\langle x, y \rangle) = x$, second projection of pairing: $\pi_2(\langle x, y \rangle) = y$

We will use pairs to construct triples: $\langle x, y, z \rangle \equiv \langle x, \langle y, z \rangle \rangle$. From now on, constant symbols as h_i and R_i will be used as meta-symbols, they are not the actual elements of $\mathcal{N}$ or $\mathcal{H}$.

We define the roles of principals as follows: the initiator, communicating with intended party Q, does the following sequence of steps in session i (denoted by $Init^A_{NSL}[A, i, Q, N_1, h_1, h_3, R_1, R_3]$):

1. Receives handle h_1 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_3, 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 $c_i\big(A, Q, N_1, \pi_1(\pi_2(dec(h_3, dK_A)))\big)$.

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 denote informally 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 $c_r(\pi_2(dec(h_2, dK_B)), B, \pi_1(dec(h_2, dK_B)), N_2)$.

▶ **Example 3.1.** We show the beginning of a possible branch in the symbolic execution of a single session of the NSL protocol.

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

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 the 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.

- $\phi_0 = \nu_{K_A K_B AB}()$, $\Theta_0 = \emptyset$
- $H_1 = \emptyset$, $\phi_1 = \nu_{K_A K_B AB}(p_1 \mapsto (A, B, eK_A, eK_B))$, $\Theta_1 = \emptyset$
- $H_2 = \langle h_1 \rangle$, ϕ_2 extends ϕ_1 with $p_1 \mapsto \{\langle N_1, A\rangle\}^{R_1}_{eK_B}$, $\Theta_2 = \{\phi_1 \vdash h_1\}$
- $H_3 = \langle h_1, h_2 \rangle$, 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 3.2.** Consider again Example 3.1, and 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: $\hat{\phi}, \vec{x}, x \vdash x$
  - Increasing capabilities: $\hat{\phi}, \vec{x} \vdash y \longrightarrow \hat{\phi}, \vec{x}, x \vdash y$
  - Commutativity: If $\vec{x}'$ is a permutation of $\vec{x}$, then $\hat{\phi}, \vec{x} \vdash y \longrightarrow \hat{\phi}, \vec{x}' \vdash y$
  - Transitivity of derivability: $\hat{\phi}, \vec{x} \vdash \vec{y} \wedge \hat{\phi}, \vec{x}, \vec{y} \vdash \vec{z} \longrightarrow \hat{\phi}, \vec{x} \vdash \vec{z}$
  - Functions are derivable: $\hat{\phi}, \vec{x} \vdash f(\vec{x})$

- **Axioms for Randomly Generated Items.** These are relations involving $\mathsf{RandGen}()$ 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). No telepathy expresses that randomly generated items not yet sent out are not guessable. It is sound as 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. The condition $\vec{x}, y \preccurlyeq \hat{\phi}$ ensures that $\vec{x}, y$ do not contain future knowledge of the adversary in the form of handles computed later.
  - 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:

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

- **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-CCA2 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.

▶ **Theorem 4.1** (Secrecy of CCA2 Encryption). *If the encryption scheme is IND-CCA2, then the following formula is computationally sound.*

$$\mathsf{RandGen}(K) \wedge eK \sqsubseteq \hat{\phi} \wedge \mathsf{fresh}(R; \hat{\phi}, \vec{x}, x, y) \wedge \vec{x}, x, y \preccurlyeq \hat{\phi} \wedge \hat{\phi}, \vec{x}, \{x\}_{eK}^R \vdash y$$
$$\longrightarrow dK \sqsubseteq_{\not{d}} \hat{\phi}, \vec{x}, x \vee \hat{\phi}, \vec{x} \vdash y$$

This axiom is a stronger version of the one proved in [7] and its proof is available in the full version of our paper [5]. It says that if K was correctly generated, R is fresh, and y can be derived with the help of $\{x\}_{eK}^R$, then it can be derived without $\{x\}_{eK}^R$, or dK has been sent out or it is in $\vec{x}$ or x. The condition $dK \not\sqsubseteq_{\not{d}} \hat{\phi}, \vec{x}, x$ (if moved to the premise) ensures dK has not been revealed, and it is also an easy way to avoid key-cycles, sufficient for the NSL protocol. A tighter axiom is left for future work. Again, $\vec{x}, x, y \preccurlyeq \hat{\phi}$ ensures that $\vec{x}, x, y$ do not contain future information in the form of handles not computable from ϕ. 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. For proving this axiom we need the decryption oracle in the CCA2 game *before* the ciphertext was created by the encryption oracle, but not after.
Let us now consider the case of non-malleability and suppose that 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 $\mathsf{maycorrupt}_{\mathrm{CCA2}}(u; \hat{\phi}, \vec{x})$ be a constraint meaning that u is a term that is paired-together all terms which occur in $\phi, \vec{x}$ not guarded by an honest encryption, immediately under one of the functions $f_1, \ldots, f_n$ or immediately under an honest decryption, not in the key position.

▶ **Theorem 4.2** (Non-Malleability of CCA2 Encryption). *If the encryption scheme is IND-CCA2, then the following formula is computationally sound.*

$$\begin{array}{l} \exists u(\mathsf{maycorrupt}_{\mathrm{CCA2}}(u;\hat{\phi},\vec{x}) \wedge \hat{\phi},\vec{x},u \not\vdash N) \ \wedge \ \mathsf{RandGen}(N) \ \wedge \ \mathsf{RandGen}(K) \\ \wedge \ eK \sqsubseteq \hat{\phi} \ \wedge \ \vec{x} \preccurlyeq \hat{\phi} \ \wedge \ N \sqsubseteq \hat{\phi},\vec{x} \ \wedge \ \hat{\phi},\vec{x} \vdash y \ \wedge \ \hat{\phi},\vec{x}, dec(y,dK) \vdash N \\ \rightarrow \ \exists K'(\mathsf{RandGen}(K') \wedge dK' \sqsubseteq_{\not{d}} \hat{\phi},\vec{x}) \vee \hat{\phi},\vec{x} \vdash N \vee \exists x R(y = \{x\}^R_{eK} \wedge \{x\}^R_{eK} \sqsubseteq \hat{\phi},\vec{x}) \end{array}$$

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 some 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 conjunct 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. Also, there is no guarantee for a CCA2 encryption that, for example, $dec(N, dK)$ does not decrypt to N. In the NSL case, $\mathsf{maycorrupt}_{\mathrm{CCA2}}$ only refers to decryptions as we do not have other function symbols. The proof is in [5].

- 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: $\mathsf{RandGen}(N) \wedge \hat{\phi},\vec{x},c(x,y,z,w) \vdash N \rightarrow \hat{\phi},\vec{x} \vdash N$
  - c cannot be forged and cannot be subpart of a term:
    $\hat{\phi},\vec{x} \vdash c(x,y,z,w) \longrightarrow c(x,y,z,w) \sqsubseteq \hat{\phi} \vee x_1 = c(x,y,z,w) \vee \ldots \vee x_l = c(x,y,z,w)$
  - 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,
  - $\mathsf{RandGen}(N) \rightarrow \neg W(\pi_2(N))$.

  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 Section 5. 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 supposed to be 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. This means that *security of the NSL protocol does not require tagging of nonces, pairs, encryptions.* This axiom is used in step in 2.2.2 of the full proof of the NSL protocol presented in [5].

5 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 that $\langle n, Q\rangle = N$. To show that this is not at all unrealistic, suppose that 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 $1/2^8$, i.e. 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 between A and B, two honest agents.
3. The adversary, acting as agent Q initiates a new session with B, sending $\{N_2\}_B$ to him.
4. Since B believes this is a new session with Q, it will parse $\{N_2\}_B$ 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 $1/2^8$.
5. B then 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.

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

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. For example, the reader may suggest that this attack is in fact very similar to the one shown in [19] (as we both wrote it as $N = \langle n, Q\rangle$). However, there is a very fundamental difference. The attack in [19] is based on the fact that an honest agent sends a pair with a nonce and an agent name, and the receiving honest agent understands this as a single nonce. In other words, in [19] the honest receiver reads the pair of a nonce and a name into an input variable meant for a nonce. There, n is the honest nonce and N is the input variable. In our attack, it is an actual nonce that is understood by the receiver as a pair of a nonce and a name. In our case, an actual nonce is read into the pair of two input variables: one for a nonce and another for a name. Here N is the honest nonce and n corresponds to the input variable. This is a fundamental difference as in our case there are no atomic objects at all. Even an honest nonce is allowed to be split. To our knowledge, this is the first such attack on the NSL protocol.

6 Examples for Proving Inconsistency

Here we look at three small example proofs so that the reader can become familiar with 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.*

▶ **Example 6.1.** We start with a very trivial example. It is rather obvious that in the execution of the NSL protocol in Example 3.1, $\phi_2 \not\vdash A$ should be inconsistent with the axioms as A is included in ϕ_2. We can derive it the following way: observe that

$$\phi_2 \equiv A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \equiv \phi_0, A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B}. \tag{1}$$

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, $\phi_0, A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash A$, so its negation is inconsistent with the axioms. Hence, $\mathcal{M}, \sigma \not\models A, B, eK_A, eK_B, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \not\vdash A$ for any model $\mathcal{M}$, which implies by (1) that $\phi_2 \not\vdash A$ is inconsistent with the axioms.

▶ **Example 6.2.** 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, in order to get a contradiction, that this is not the case, *i.e.*, $\phi_1, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash N_1$. To apply the secrecy axiom consider that $\vec{x} = \langle\rangle, x = \langle N_1, A\rangle$, and $y = N_1$. Since K_B was correctly generated (appeared as a name), $\mathsf{RandGen}(K_B)$ holds. By Example 3.1 we have $eK_B \sqsubseteq \phi_1$. $\mathsf{fresh}(R_1; \phi_1, \vec{x}, x, y)$ 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 $\vec{x} \preccurlyeq \phi_1$, $x \preccurlyeq \phi_1$ and $y \preccurlyeq \phi_1$ because none has handles, $\phi_1, \{\langle N_1, A\rangle\}^{R_1}_{eK_B} \vdash N_1$ was supposed, and $dK_B \not\sqsubseteq \phi_1, \vec{x}, x$, one may apply the secrecy axiom and get $\phi_1 \vdash N_1$. At Step 1 we have $\mathsf{fresh}(N_1; \phi_1)$ as $\mathsf{RandGen}(N_1) \wedge N_1 \not\sqsubseteq \phi_1$, so $\mathsf{fresh}(N_1; \phi_1) \wedge \phi_1 \vdash N_1$ holds, which contradicts the no telepathy axiom.

▶ **Example 6.3.** 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. 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 6.1 it also follows that for any m, the axioms imply that $\phi_m \vdash A$: it is clear from Example 6.1, that $\phi_2 \vdash A$ is implied, then from the above, by induction, $\phi_m \vdash A$ is also implied. Note that the induction is not within FOL, we used the induction on the number of execution steps.

7 Correctness Proof of NSL

We present now the correctness of the NSL protocol for any (bounded) number of sessions. 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 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 messages sent by A and B, show that $\phi_m \not\vdash N$ together with the axioms and agent checks imply $\phi_{m+1} \not\vdash N$ for each possible sent message. Hence, $\phi_m \not\vdash N$, 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 nonces remain secret, together with non-malleability, we show that the initiator also finished his task and the corresponding values seen by the two parties 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. The fact that N is a nonce sent by A to B in the NSL protocol can be expressed as $\exists R\big(\{N, A\}^R_{eK_B} \sqsubseteq \hat{\phi}\big)$. If B sent it to A, that means $\exists hR\big(\{\pi_1\left(dec(h, dK_B)\right), N, B\}^R_{eK_A} \sqsubseteq \hat{\phi}\big)$. So, such nonces can be characterized by the condition

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

Then the secrecy property we want to show is that $\forall N\big(C[N] \longrightarrow \hat{\phi} \not\vdash N\big)$, meaning that such nonces cannot be computed by the adversary. It is equivalent to show that its negation, $\exists N\big(C[N] \wedge \hat{\phi} \vdash N\big)$, 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. If $\vec{x}$ is a list of a finite number of nonces $\vec{x} \equiv N_1, ..., N_l$ that were all generated by either A or B (possibly intended to each other, possibly intended for other possibly malicious agents), and they are all different from N, then we say condition $C'[\vec{x}, N]$ is satisfied:

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

We will carry out an inductive proof on the length of ϕ_n. As it turns out, in order to avoid loops in the proof, instead of $\exists N\big(C[N] \wedge \hat{\phi} \vdash N\big)$, it is better to prove that

$$\exists N \exists \vec{x}\big(C[N] \wedge C'[\vec{x}, N] \wedge \hat{\phi}, \vec{x} \vdash N\big) \tag{2}$$

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

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

is inconsistent with the axioms and agent checks. We do this by fixing an arbitrary N satisfying $C[N]$, and by showing that if for some $m < n$, $\exists \vec{x}\big(C'[\vec{x}, N] \wedge \phi_m, \vec{x} \vdash N\big)$ is inconsistent with the axioms and agent checks, then $\exists \vec{x}\big(C'[\vec{x}, N] \wedge \phi_{m+1}, \vec{x} \vdash N\big)$ is also inconsistent with the axioms and agent checks. As at $m = 0$ the statement follows from no telepathy, we are done. To be completely precise, we would have to take into account $\mathsf{maycorrupt}_{\mathrm{CCA2}}$ as well, but this is trivial for NSL as only those things are decrypted that came from the clear (only handles are decrypted). Again, note that within C and C', $\hat{\phi}$ is always ϕ_n and not ϕ_m.

▶ **Proposition 7.1.** *In the above execution of NSL protocol, let N be such that $C[N]$ is satisfied, and let $m < n$. If for all $\vec{x}$ such that $C'[\vec{x}, N]$ holds, the axioms and agent checks imply (by FOL deduction rules) that $\phi_m, \vec{x} \not\vdash N$, then for all $\vec{x}$ such that $C'[\vec{x}, N]$ holds, the axioms and agent checks imply (by FOL deduction rules) that $\phi_{m+1}, \vec{x} \not\vdash N$ holds.*

The proof is in [5]. Once this is shown, we still have to prove that the property initially holds, that is, $\exists N \exists \vec{x}\big(C[N] \wedge C'[\vec{x}, N] \wedge \phi_0, \vec{x} \vdash N\big)$ is inconsistent with the axioms. Let $C[N]$ and $C'[\vec{x}, N]$ hold for N and $\vec{x} \equiv N_1, ..., N_l$. 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, $\phi_0, N_1 \not\vdash N$. 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 $\exists N \exists \vec{x}\big(C[N] \wedge C'[\vec{x}, N] \wedge \phi_0, \vec{x} \vdash N\big)$ is indeed inconsistent with the axioms. Therefore, together with the induction step of Proposition 7.1, we have:

▶ **Theorem 7.2** (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 any* $n \in \mathbb{N}$ *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 never broken. That is, nonces that were generated by A or B and intended to be sent between each other, remain secret. In particular, asking $\vec{x}$ 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

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

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

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

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, as the following theorem states with the proof available in [5].

▶ **Theorem 7.3** (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*

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

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*, pages 220–230. ACM, 2003.

4 M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

5 G. Bana, P. Adão, and H. Sakurada. Computationally Complete Symbolic Attacker in Action—Long version, 2012. Available at `http://eprint.iacr.org/2012/316`.

6 G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker, 2012. Available at `http://eprint.iacr.org/2012/019`.

7 G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In *POST'12*, volume 7215 of *LNCS*, pages 189–208. Springer, 2012.

8 G. Bana, K. Hasebe, and M. Okada. Computational Semantics for First-Order Logical Analysis of Cryptographic Protocols. In *Formal to Practical Security*, volume 5458 of *LNCS*, pages 33–56. Springer, 2007.

9 G. Bana, K. Hasebe, and M. Okada. Secrecy-oriented first-order logical analysis of cryptographic protocols, 2010. Available at `http://eprint.iacr.org/2010/080`.

10 G. Bana, P. Mohassel, and T. Stegers. Computational Soundness of Formal Indistinguishability and Static Equivalence. In *ASIAN'06*, volume 4435 of *LNCS*, pages 182–196. Springer, 2007.

11 G. Barthe, B. Grégoire, and S. Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *POPL'09*, pages 90–101. ACM, 2009.

12 M. Baudet, V. Cortier, and S. Kremer Computationally Sound Implementations of Equational Theories Against Passive Adversaries. In *ICALP'05*, volume 3580 of *LNCS*, pages 652–663. 2005.

13 B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, 2008.

14 H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *CCS'08*, pages 109–118. ACM, 2008.

15 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 *LIPIcs*, pages 29–44. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

16 V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *ESOP'05*, volume 3444 of *LNCS*, pages 157–171. Springer, 2005.

17 V. Cortier and B. Warinschi. A composable computational soundness notion. In *CCS'11*, pages 63–74. ACM, 2011.

18 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. 2005.

19 J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS'01*, pages 166–175. ACM, 2001.

20 B. Warinschi. A computational analysis of the Needham-Schroeder protocol. In *CSFW'03*, pages 248–262. IEEE Computer Society, 2003.